Music Classification - Training and Testing

Makenna Barton - bartonmj@uw.edu

March 2020

Abstract This report outlines the techniques and procedures necessary to complete a supervised machine learning problem. Below describes the steps taken to train classifiers to recognize different music artists or music genres. The mathematical techniques and the statistical analysis of the data used to establish thresholds between each class of data are explored. The three different classifiers are tested with new music files to determine their accuracy in predicting an artist or genre.

1 Introduction and Overview

Three separate music classifications will be tested: three artists from different genres, three artists from the same genres, and three genres consisting of music from three artists within each genre. The genres I have selected are classical, country, and rock. The classical artists used are Bach, Mozart, and Yo-Yo Ma. The country artists are Dierks Bentley, Kenny Chesney, and Luke Combs. The rock artists are the bands AC/DC, Aerosmith, and The Rolling Stones.

Classification 1 (comparing artists across genres) will compare Bach, Luke Combs, and the Rolling Stones.

Classification 2 will test the rock genre and compare AC/DC, Aerosmith, and The Rolling Stones to each other.

Classification 3 will compare the genres holistically, each genre including songs from the three artists in the genre as mentioned above.

The training dataset for this supervised machine learning problem consisted of 30 songs per artist, totalling to 90 songs per genre. Once testing the classifiers, a new set of data was collected consisting of 7 songs per artist, therefore a set of 21 songs to test against classifier 1 and 2 and a set of 63 songs to test classifier 3 with.

2 Theoretical Background

Similar to the previous assignment, solving this problem relies heavily on the processes of Singular Value Decomposition/Principal Component Analysis. To recall, the reduced SVD is written mathematically as:

$$A = \hat{U}\hat{\Sigma}V^T \tag{1}$$

where $\hat{U}$ and V^T are unitary matrices that contain information on rotation and $\hat{\Sigma}$ is a diagonal matrix describing the stretch in each direction. The columns of $\hat{U}$ are the left singular vectors of A and the columns of V are called the right singular vectors of A. Each non-zero value in the $\hat{\Sigma}$ matrix is a singular value of A and the total number of non-zero singular values is the rank of A.

To make statistical sense out of the Singular Value Decomposition, this problem uses Linear Discriminant Analysis (LDA). As defined by Nathan Kutz, the goal of LDA is two things: "find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data". Mathematically, this goal is written:

$$w = \arg\max_w \frac{w^T S_B w}{w^T S_W w} \tag{2}$$

with the between-class scatter matrix, S_B, and within-class scatter matrix, S_W, written as:

$$S_B = \sum_{j=1}^{n} m_j (\vec{\mu_j} - \vec{\mu})(\vec{\mu_j} - \vec{\mu})^T \tag{3}$$

$$S_W = \sum_{j=1}^{n} \sum_{\vec{x}} (\vec{x} - \vec{\mu_j})(\vec{x} - \vec{\mu_j})^T \tag{4}$$

where m_j is the number of samples in class j, μ is the mean of all the data, and μ_j is the mean of the data in class j. Using these matrices we can find the solution to Equation (2) with a generalized eigenvalue problem:

$$S_B w = \lambda S_W w \tag{5}$$

The information provided by Linear Discriminant Analysis and solving this eigenvalue problem gives us the quantity of interest and the projection basis. The projections are what enable us to separate the different classes of data into categories mathematically.

3 Algorithm Implementation and Development

The first step of this process was collecting music clips from the chosen artists in the three genres. I collected 30 5-second music clips from each artist to train my classifiers with. After loading in the data and reorganizing it and reshaping it into matrices representing each artist I was able to being to process them. I created spectrograms for each of the 30 songs as the first step of my analysis. I analyzed the songs through their spectrograms instead of their raw signal because spectrograms are a way of determining the differences in signals. I then created a training function that contained all of the processing of the song data (in the form of spectograms) to allow me to use it for the different classification situations. In the training function, I first computed the SVD on the matrices of the song spectrograms for the three artists/genres. Then I used PCA to get all of the projections of the songs onto the principal components as calculated by the SVD.

After separating the songs for the three artists or three genres, I begin Linear Discriminant Analysis. I calculate their means to compute the within-class variance and between-class variance. Using these variances I am able to calculate the Linear Discriminant Analysis. I solve the eigenvalue problem to get the projection basis and project the songs from each artist/genre onto the basis.

I finally order and sort the data to establish thresholds to separate the data in each genre or from each artist.

Later on, when testing the accuracy of classifiers on predicting the artist or genre of a song, the steps for processing the data were the following: compute the spectrogram, project the data onto the SVD modes, project this decomposition onto the projection basis as determined by the LDA of that classifier, then compare the values to the set thresholds.

4 Computational Results

The graphs in Figure 1 show the song data plotted after training the classifiers for each situation. Figure 2 is an example of how much of the data from each artist in the first classifier was captured within the calculated thresholds. After testing 21 random music clips (7 songs from each artist) in the first classifier (using 15 features/dimension), it correctly predicted the artist of 17 of them, reporting approximately 81% accuracy.

Testing the second classifier (using 20 features/dimensions) with 21 music clips (7 from each artist), the classifier was able to correctly predict the artist of 8 songs, 40% accuracy rate.

Finally, testing the third classifier (using 20 features/dimensions), testing genres, with 63 random samples of music (21 from each genre), the classifier correctly predicted the genre of 39 of them, approximately a 62% accuracy rate. It should be noted for this test, the classifier correctly predicted 18/21 of the songs in the classical genre and 21/21 songs in the rock genre.

5 Summary and Conclusions

In this problem we used existing data sets to perform supervised machine learning to train music classifiers, that classify by artist or genre. Linear Discriminant Analysis is a statistical tool that results in a projection basis that minimizes intra-class variance while maximizing the distance between inter-class data. LDA helped us to set thresholds to determine the artist or genre a given song belongs to. After testing my classifiers, I found the highest successful prediction rate from classifier 1, three artists from different genres. Based on my knowledge of the classifiers and LDA, I believe we saw the greatest accuracy in this classifier due to the diversity of the data. The more "different" each class of data being analyzed, the easier it is to distinguish the groups from each other using LDA. The worst accuracy of a classifier was in the second classifier that compared three artists within the same genre. The lack of success in this classifier came from the same reasons for the success in the first classifier, how

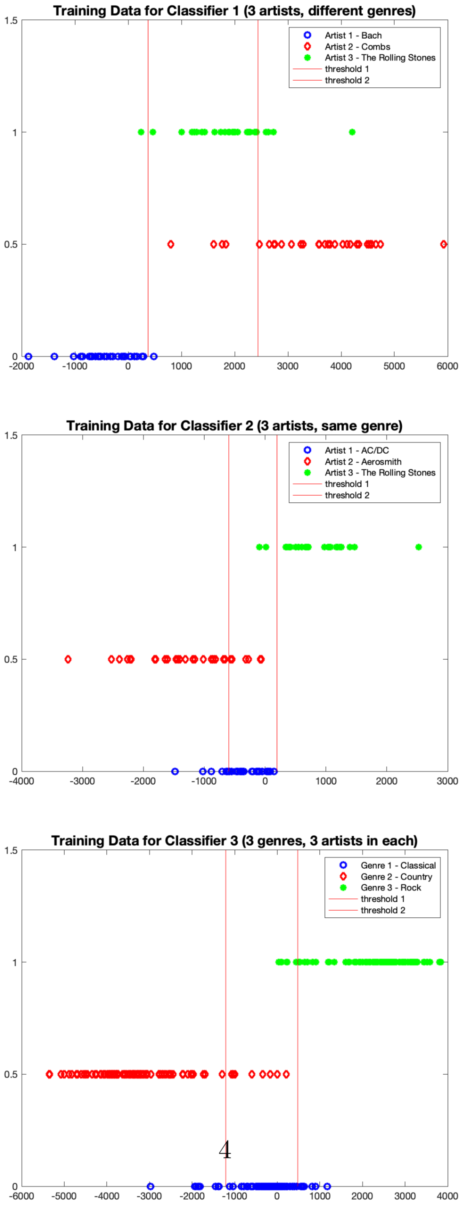

Figure 1: Above are the projected data and statistically decided thresholds for each classifier.

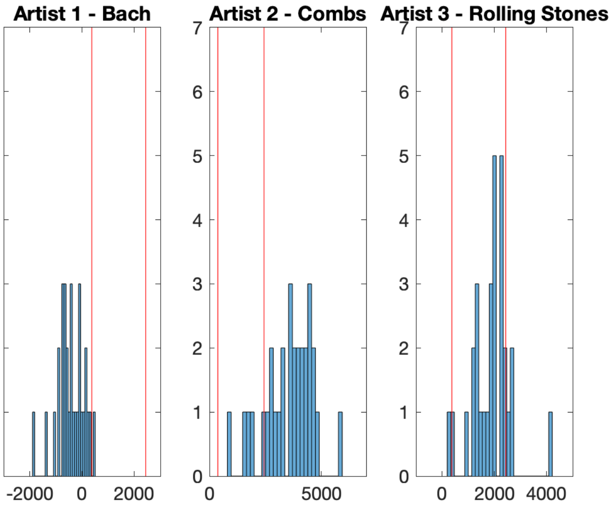

Figure 2: Above shows the amount of song data that was captured in each section of the Classifier 1.

similar the data was. I think it is important to note the success of the third classifier when looking specifically at the Classical and Rock genres. The use of more training data could be one reason this classifier was more accurate distinguishing between classical and rock music, but I also think that may have contributed to the confusion the classifier had between the rock and country genres. When compared in a more general sense, rock music and country music share many elements that make them harder to distinguish.

6 Appendix A - MATLAB functions used and brief implementation explanation

`svd()`- Singular Value Decomposition, outputs the factorization into a left singular vector matrix, a matrix with the singular values, and a matrix with the right singular values
`spectrogram()`- returns the short-time Fourier transform of the input signal. Each column of the outputted matrix contains an estimate of the short-term, time-localized frequency content of the input.

7 Appendix B - MATLAB Codes

Matlab Code

```
%% Test 1 - Luke Combs vs Bach vs Rolling Stones
clear; clc; close all;

[Combs(:,1),Fs] = audioread('Combs1.wav');
[Combs(:,2),Fs] = audioread('Combs2.wav');
[Combs(:,3),Fs] = audioread('Combs3.wav');
[Combs(:,4),Fs] = audioread('Combs4.wav');
[x5,Fs] = audioread('Combs5.wav');
```

```
[Combs(:,6),Fs] = audioread('Combs6.wav');
[x7,Fs] = audioread('Combs7.wav');
[x8,Fs] = audioread('Combs8.wav');
[x9,Fs] = audioread('Combs9.wav');
[x10,Fs] = audioread('Combs10.wav');
[x11,Fs] = audioread('Combs11.wav');
[Combs(:,12),Fs] = audioread('Combs12.wav');
[x13,Fs] = audioread('Combs13.wav');
[Combs(:,14),Fs] = audioread('Combs14.wav');
[Combs(:,15),Fs] = audioread('Combs15.wav');
[Combs(:,16),Fs] = audioread('Combs16.wav');
[Combs(:,17),Fs] = audioread('Combs17.wav');
[Combs(:,18),Fs] = audioread('Combs18.wav');
[Combs(:,19),Fs] = audioread('Combs19.wav');
[Combs(:,20),Fs] = audioread('Combs20.wav');
[Combs(:,21),Fs] = audioread('Combs21.wav');
[x22,Fs] = audioread('Combs22.wav');
[Combs(:,23),Fs] = audioread('Combs23.wav');
[Combs(:,24),Fs] = audioread('Combs24.wav');
[Combs(:,25),Fs] = audioread('Combs25.wav');
[Combs(:,26),Fs] = audioread('Combs26.wav');
[Combs(:,27),Fs] = audioread('Combs27.wav');
[Combs(:,28),Fs] = audioread('Combs28.wav');
[Combs(:,29),Fs] = audioread('Combs29.wav');
[Combs(:,30),Fs] = audioread('Combs30.wav');
Combs = Combs(1:220160,:);
Combs(:,5) = x5(1:220160);
Combs(:,7) = x7(1:220160);
Combs(:,8) = x8(1:220160);
Combs(:,9) = x9(1:220160);
Combs(:,10) = x10(1:220160);
Combs(:,11) = x11(1:220160);
Combs(:,13) = x13(1:220160);
Combs(:,22) = x22(1:220160);

[Bach(:,1),Fs] = audioread('Bach1.wav');
[Bach(:,2),Fs] = audioread('Bach2.wav');
[Bach(:,3),Fs] = audioread('Bach3.wav');
[x4,Fs] = audioread('Bach4.wav');
[Bach(:,5),Fs] = audioread('Bach5.wav');
[x6,Fs] = audioread('Bach6.wav');
[Bach(:,7),Fs] = audioread('Bach7.wav');
[x8,Fs] = audioread('Bach8.wav');
[Bach(:,9),Fs] = audioread('Bach9.wav');
[x10,Fs] = audioread('Bach10.wav');
[Bach(:,11),Fs] = audioread('Bach11.wav');
```

```
[x12,Fs] = audioread('Bach12.wav');
[Bach(:,13),Fs] = audioread('Bach13.wav');
[x14,Fs] = audioread('Bach14.wav');
[Bach(:,15),Fs] = audioread('Bach15.wav');
[Bach(:,16),Fs] = audioread('Bach16.wav');
[Bach(:,17),Fs] = audioread('Bach17.wav');
[Bach(:,18),Fs] = audioread('Bach18.wav');
[Bach(:,19),Fs] = audioread('Bach19.wav');
[Bach(:,20),Fs] = audioread('Bach20.wav');
[Bach(:,21),Fs] = audioread('Bach21.wav');
[Bach(:,22),Fs] = audioread('Bach22.wav');
[Bach(:,23),Fs] = audioread('Bach23.wav');
[x24,Fs] = audioread('Bach24.wav');
[Bach(:,25),Fs] = audioread('Bach25.wav');
[Bach(:,26),Fs] = audioread('Bach26.wav');
[Bach(:,27),Fs] = audioread('Bach27.wav');
[Bach(:,28),Fs] = audioread('Bach28.wav');
[Bach(:,29),Fs] = audioread('Bach29.wav');
[Bach(:,30),Fs] = audioread('Bach30.wav');
Bach = Bach(1:221184,:);
Bach(:,4) = x4(1:221184);
Bach(:,6) = x6(1:221184);
Bach(:,8) = x8(1:221184);
Bach(:,10) = x10(1:221184);
Bach(:,12) = x12(1:221184);
Bach(:,14) = x14(1:221184);
Bach(:,24) = x24(1:221184);

[x1,Fs] = audioread('Stones1.wav');
[Stones(:,2),Fs] = audioread('Stones2.wav');
[Stones(:,3),Fs] = audioread('Stones3.wav');
[Stones(:,4),Fs] = audioread('Stones4.wav');
[Stones(:,5),Fs] = audioread('Stones5.wav');
[Stones(:,6),Fs] = audioread('Stones6.wav');
[Stones(:,7),Fs] = audioread('Stones7.wav');
[x8,Fs] = audioread('Stones8.wav');
[Stones(:,9),Fs] = audioread('Stones9.wav');
[Stones(:,10),Fs] = audioread('Stones10.wav');
[x11,Fs] = audioread('Stones11.wav');
[Stones(:,12),Fs] = audioread('Stones12.wav');
[x13,Fs] = audioread('Stones13.wav');
[Stones(:,14),Fs] = audioread('Stones14.wav');
[Stones(:,15),Fs] = audioread('Stones15.wav');
[Stones(:,16),Fs] = audioread('Stones16.wav');
[x17,Fs] = audioread('Stones17.wav');
[x18,Fs] = audioread('Stones18.wav');
```

```
[x19,Fs] = audioread('Stones19.wav');
[x20,Fs] = audioread('Stones20.wav');
[Stones(:,21),Fs] = audioread('Stones21.wav');
[x22,Fs] = audioread('Stones22.wav');
[Stones(:,23),Fs] = audioread('Stones23.wav');
[Stones(:,24),Fs] = audioread('Stones24.wav');
[Stones(:,25),Fs] = audioread('Stones25.wav');
[x26,Fs] = audioread('Stones26.wav');
[x27,Fs] = audioread('Stones27.wav');
[x28,Fs] = audioread('Stones28.wav');
[x29,Fs] = audioread('Stones29.wav');
[x30,Fs] = audioread('Stones30.wav');
Stones = Stones(1:220672,:);
Stones(:,1) = x1(1:220672);
Stones(:,8) = x8(1:220672);
Stones(:,11) = x11(1:220672);
Stones(:,13) = x13(1:220672);
Stones(:,17) = x17(1:220672);
Stones(:,18) = x18(1:220672);
Stones(:,19) = x19(1:220672);
Stones(:,20) = x20(1:220672);
Stones(:,22) = x22(1:220672);
Stones(:,26) = x26(1:220672);
Stones(:,27) = x27(1:220672);
Stones(:,28) = x28(1:220672);
Stones(:,29) = x29(1:220672);
Stones(:,30) = x30(1:220672);

Stones = Stones(1:220160,:);
Bach = Bach(1:220160,:);
feature = 15;

SpecBach = zeros(262152,30);
SpecCombs = zeros(262152,30);
SpecStones = zeros(262152,30);
for jj = 1:30
    S1 = spectrogram(Bach(:,jj));
    S2 = spectrogram(Combs(:,jj));
    S3 = spectrogram(Stones(:,jj));
    SpecBach(:,jj) = abs(S1(:));
    SpecCombs(:,jj) = abs(S2(:));
    SpecStones(:,jj) = abs(S3(:));
end

[U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
    test1_trainer(SpecBach,SpecCombs,SpecStones,feature);
```

```

[Test(:,1),Fs] = audioread('BachTest1.wav');
[Test(:,2),Fs] = audioread('BachTest2.wav');
[Test3,Fs] = audioread('BachTest3.wav');
[Test4,Fs] = audioread('BachTest4.wav');
[Test5,Fs] = audioread('BachTest5.wav');
[Test6,Fs] = audioread('BachTest6.wav');
[Test7,Fs] = audioread('BachTest7.wav');
[Test(:,8),Fs] = audioread('CombsTest1.wav');
[Test(:,9),Fs] = audioread('CombsTest2.wav');
[Test10,Fs] = audioread('CombsTest3.wav');
[Test11,Fs] = audioread('CombsTest4.wav');
[Test12,Fs] = audioread('CombsTest5.wav');
[Test13,Fs] = audioread('CombsTest6.wav');
[Test14,Fs] = audioread('CombsTest7.wav');
[Test15,Fs] = audioread('StonesTest1.wav');
[Test16,Fs] = audioread('StonesTest2.wav');
[Test17,Fs] = audioread('StonesTest3.wav');
[Test18,Fs] = audioread('StonesTest4.wav');
[Test19,Fs] = audioread('StonesTest5.wav');
[Test20,Fs] = audioread('StonesTest6.wav');
[Test21,Fs] = audioread('StonesTest7.wav');
Test = Test(1:220160,:);
Test(:,3) = Test3(1:220160);
Test(:,4) = Test4(1:220160);
Test(:,5) = Test5(1:220160);
Test(:,6) = Test6(1:220160);
Test(:,7) = Test7(1:220160);
Test(:,10) = Test10(1:220160);
Test(:,11) = Test11(1:220160);
Test(:,12) = Test12(1:220160);
Test(:,13) = Test13(1:220160);
Test(:,14) = Test14(1:220160);
Test(:,15) = Test15(1:220160);
Test(:,16) = Test16(1:220160);
Test(:,17) = Test17(1:220160);
Test(:,18) = Test18(1:220160);
Test(:,19) = Test19(1:220160);
Test(:,20) = Test20(1:220160);
Test(:,21) = Test21(1:220160);

SpecTest = zeros(262152,21);
for jj = 1:21
    S1 = spectrogram(Test(:,jj));
    SpecTest(:,jj) = abs(S1(:));
end
```

```
%%
TestMatrix = U'*SpecTest;
projection = w'*TestMatrix;
True = [0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1;
    0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0];
ResVec(1,:) = (projection>threshold1);
ResVec(2,:) = (projection>threshold2)
True-ResVec
%% Plot artist1/2/3 projections onto w
figure(1)
p = plot(sortart1,zeros(30),'ob','Linewidth',2)
for j = 2:30
set(get(get(p(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
hold on
p1 = plot(sortart2,.5*ones(30),'dr','Linewidth',2)
for j = 2:30
set(get(get(p1(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
p2 = plot(sortart3,ones(30),'*g','Linewidth',2)
for j = 2:30
set(get(get(p2(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
plot([threshold1 threshold1],[0 10],'r')
plot([threshold2 threshold2],[0 10],'r')
ylim([0 1.5])
title('Training Data for Classifier 1 (3 artists, different genres)'...
    ,'Fontsize',14)
legend('Artist 1 - Bach','Artist 2 - Combs',...
    'Artist 3 - The Rolling Stones','threshold 1','threshold 2')
print(gcf,'Test1_dataplot.png','-dpng')
%% Plot histogram of training data with thresholds
figure(5)
subplot(1,3,1)
histogram(sortart1,30);
hold on, plot([threshold1 threshold1],[0 7],'r')
plot([threshold2 threshold2],[0 7],'r')
set(gca,'Xlim',[-3000 3000],'Ylim',[0 7],'Fontsize',14)
title('Artist 1 - Bach')
subplot(1,3,2)
histogram(sortart2,30);
hold on,
plot([threshold1 threshold1],[0 7],'r')
plot([threshold2 threshold2],[0 7],'r')
set(gca,'Xlim',[0 7000],'Ylim',[0 7],'Fontsize',14)
title('Artist 2 - Combs')
subplot(1,3,3)
histogram(sortart3,30);
hold on, plot([threshold1 threshold1],[0 7],'r')
plot([threshold2 threshold2],[0 7],'r')
set(gca,'Xlim',[-1000 5000],'Ylim',[0 7],'Fontsize',14)
title('Artist 3 - Rolling Stones')
print(gcf,'Test1_histogram.png','-dpng')
%% Test 2 - AC/DC vs Rolling Stones vs Aerosmith
clear; clc;
[ACDC(:,1),Fs] = audioread('ACDC1.wav');
[ACDC(:,2),Fs] = audioread('ACDC2.wav');
[ACDC(:,3),Fs] = audioread('ACDC3.wav');
[ACDC(:,4),Fs] = audioread('ACDC4.wav');
[ACDC(:,5),Fs] = audioread('ACDC5.wav');
[ACDC(:,6),Fs] = audioread('ACDC6.wav');
[ACDC(:,7),Fs] = audioread('ACDC7.wav');
[z8,Fs] = audioread('ACDC8.wav');
[z9,Fs] = audioread('ACDC9.wav');
[z10,Fs] = audioread('ACDC10.wav');
[ACDC(:,11),Fs] = audioread('ACDC11.wav');
[z12,Fs] = audioread('ACDC12.wav');
[z13,Fs] = audioread('ACDC13.wav');
[ACDC(:,14),Fs] = audioread('ACDC14.wav');
[ACDC(:,15),Fs] = audioread('ACDC15.wav');
[ACDC(:,16),Fs] = audioread('ACDC16.wav');
[z17,Fs] = audioread('ACDC17.wav');
[ACDC(:,18),Fs] = audioread('ACDC18.wav');
[ACDC(:,19),Fs] = audioread('ACDC19.wav');
[ACDC(:,20),Fs] = audioread('ACDC20.wav');
[z21,Fs] = audioread('ACDC21.wav');
[ACDC(:,22),Fs] = audioread('ACDC22.wav');
[ACDC(:,23),Fs] = audioread('ACDC23.wav');
[z24,Fs] = audioread('ACDC24.wav');
[ACDC(:,25),Fs] = audioread('ACDC25.wav');
[ACDC(:,26),Fs] = audioread('ACDC26.wav');
[ACDC(:,27),Fs] = audioread('ACDC27.wav');
[z28,Fs] = audioread('ACDC28.wav');
[ACDC(:,29),Fs] = audioread('ACDC29.wav');
[ACDC(:,30),Fs] = audioread('ACDC30.wav');
ACDC = ACDC(1:220571,:);
ACDC(:,8) = z8(1:220571);
ACDC(:,9) = z9(1:220571);
ACDC(:,10) = z10(1:220571);
ACDC(:,12) = z12(1:220571);
ACDC(:,13) = z13(1:220571);
ACDC(:,17) = z17(1:220571);
ACDC(:,21) = z21(1:220571);
ACDC(:,24) = z24(1:220571);
ACDC(:,28) = z28(1:220571);

[AS(:,1),Fs] = audioread('Aerosmith1.wav');
[AS(:,2),Fs] = audioread('Aerosmith2.wav');
[AS(:,3),Fs] = audioread('Aerosmith3.wav');
[AS(:,4),Fs] = audioread('Aerosmith4.wav');
[AS(:,5),Fs] = audioread('Aerosmith5.wav');
[AS(:,6),Fs] = audioread('Aerosmith6.wav');
[AS(:,7),Fs] = audioread('Aerosmith7.wav');
[AS(:,8),Fs] = audioread('Aerosmith8.wav');
[AS(:,9),Fs] = audioread('Aerosmith9.wav');
[AS(:,10),Fs] = audioread('Aerosmith10.wav');
[AS(:,11),Fs] = audioread('Aerosmith11.wav');
[AS(:,12),Fs] = audioread('Aerosmith12.wav');
[AS(:,13),Fs] = audioread('Aerosmith13.wav');
[AS(:,14),Fs] = audioread('Aerosmith14.wav');
[AS(:,15),Fs] = audioread('Aerosmith15.wav');
[AS(:,16),Fs] = audioread('Aerosmith16.wav');
[AS(:,17),Fs] = audioread('Aerosmith17.wav');
[w18,Fs] = audioread('Aerosmith18.wav');
[AS(:,19),Fs] = audioread('Aerosmith19.wav');
[AS(:,20),Fs] = audioread('Aerosmith20.wav');
[AS(:,21),Fs] = audioread('Aerosmith21.wav');
[AS(:,22),Fs] = audioread('Aerosmith22.wav');
[AS(:,23),Fs] = audioread('Aerosmith23.wav');
[AS(:,24),Fs] = audioread('Aerosmith24.wav');
[AS(:,25),Fs] = audioread('Aerosmith25.wav');
[AS(:,26),Fs] = audioread('Aerosmith26.wav');
[AS(:,27),Fs] = audioread('Aerosmith27.wav');
[AS(:,28),Fs] = audioread('Aerosmith28.wav');
[AS(:,29),Fs] = audioread('Aerosmith29.wav');
[AS(:,30),Fs] = audioread('Aerosmith30.wav');
AS = AS(1:220160,:);
AS(:,18) = w18;

[x1,Fs] = audioread('Stones1.wav');
[Stones(:,2),Fs] = audioread('Stones2.wav');
[Stones(:,3),Fs] = audioread('Stones3.wav');
[Stones(:,4),Fs] = audioread('Stones4.wav');
[Stones(:,5),Fs] = audioread('Stones5.wav');
[Stones(:,6),Fs] = audioread('Stones6.wav');
```

```
[Stones(:,7),Fs] = audioread('Stones7.wav');
[x8,Fs] = audioread('Stones8.wav');
[Stones(:,9),Fs] = audioread('Stones9.wav');
[Stones(:,10),Fs] = audioread('Stones10.wav');
[x11,Fs] = audioread('Stones11.wav');
[Stones(:,12),Fs] = audioread('Stones12.wav');
[x13,Fs] = audioread('Stones13.wav');
[Stones(:,14),Fs] = audioread('Stones14.wav');
[Stones(:,15),Fs] = audioread('Stones15.wav');
[Stones(:,16),Fs] = audioread('Stones16.wav');
[x17,Fs] = audioread('Stones17.wav');
[x18,Fs] = audioread('Stones18.wav');
[x19,Fs] = audioread('Stones19.wav');
[x20,Fs] = audioread('Stones20.wav');
[Stones(:,21),Fs] = audioread('Stones21.wav');
[x22,Fs] = audioread('Stones22.wav');
[Stones(:,23),Fs] = audioread('Stones23.wav');
[Stones(:,24),Fs] = audioread('Stones24.wav');
[Stones(:,25),Fs] = audioread('Stones25.wav');
[x26,Fs] = audioread('Stones26.wav');
[x27,Fs] = audioread('Stones27.wav');
[x28,Fs] = audioread('Stones28.wav');
[x29,Fs] = audioread('Stones29.wav');
[x30,Fs] = audioread('Stones30.wav');
Stones = Stones(1:220672,:);
Stones(:,1) = x1(1:220672);
Stones(:,8) = x8(1:220672);
Stones(:,11) = x11(1:220672);
Stones(:,13) = x13(1:220672);
Stones(:,17) = x17(1:220672);
Stones(:,18) = x18(1:220672);
Stones(:,19) = x19(1:220672);
Stones(:,20) = x20(1:220672);
Stones(:,22) = x22(1:220672);
Stones(:,26) = x26(1:220672);
Stones(:,27) = x27(1:220672);
Stones(:,28) = x28(1:220672);
Stones(:,29) = x29(1:220672);
Stones(:,30) = x30(1:220672);

Stones = Stones(1:220160,:);
ACDC = ACDC(1:220160,:);
feature = 20;

SpecACDC = zeros(262152,30);
SpecAS = zeros(262152,30);
```

```
SpecStones = zeros(262152,30);
for jj = 1:30
    S1 = spectrogram(ACDC(:,jj));
    S2 = spectrogram(AS(:,jj));
    S3 = spectrogram(Stones(:,jj));
    SpecACDC(:,jj) = abs(S1(:));
    SpecAS(:,jj) = abs(S2(:));
    SpecStones(:,jj) = abs(S3(:));
end

[U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
test2_trainer(SpecACDC,SpecAS,SpecStones,feature);

[Test(:,1),Fs] = audioread('ACDCTest1.wav');
[Test(:,2),Fs] = audioread('ACDCTest2.wav');
[Test(:,3),Fs] = audioread('ACDCTest3.wav');
[Test(:,4),Fs] = audioread('ACDCTest4.wav');
[Test(:,5),Fs] = audioread('ACDCTest5.wav');
[Test(:,6),Fs] = audioread('ACDCTest6.wav');
[Test(:,7),Fs] = audioread('ACDCTest7.wav');
[Test(:,8),Fs] = audioread('ASTest1.wav');
[Test(:,9),Fs] = audioread('ASTest2.wav');
[Test(:,10),Fs] = audioread('ASTest3.wav');
[Test(:,11),Fs] = audioread('ASTest4.wav');
[Test(:,12),Fs] = audioread('ASTest5.wav');
[Test(:,13),Fs] = audioread('ASTest6.wav');
[Test(:,14),Fs] = audioread('ASTest7.wav');
[Test15,Fs] = audioread('StonesTest1.wav');
[Test16,Fs] = audioread('StonesTest2.wav');
[Test17,Fs] = audioread('StonesTest3.wav');
[Test(:,18),Fs] = audioread('StonesTest4.wav');
[Test19,Fs] = audioread('StonesTest5.wav');
[Test(:,20),Fs] = audioread('StonesTest6.wav');
[Test(:,21),Fs] = audioread('StonesTest7.wav');

Test = Test(1:220160,:);
Test(:,15) = Test15(1:220160);
Test(:,16) = Test16(1:220160);
Test(:,17) = Test17(1:220160);
Test(:,19) = Test19(1:220160);

SpecTest = zeros(262152,21);
for jj = 1:21
    S1 = spectrogram(Test(:,jj));
    SpecTest(:,jj) = abs(S1(:));
end
```

```
%%
TestMatrix = U'*SpecTest;
projection = w'*TestMatrix;
True = [1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1];
ResVec(1,:) = (projection>threshold1);
ResVec(2,:) = (projection>threshold2)
True-ResVec
%% Plot artist1/2/3 projections onto w
figure(2)
p = plot(sortart1,zeros(30),'ob','Linewidth',2)
for j = 2:30
set(get(get(p(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
hold on
p1 = plot(sortart2,.5*ones(30),'dr','Linewidth',2)
for j = 2:30
set(get(get(p1(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
p2 = plot(sortart3,ones(30),'*g','Linewidth',2)
for j = 2:30
set(get(get(p2(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
plot([threshold1 threshold1],[0 10],'r')
plot([threshold2 threshold2],[0 10],'r')
ylim([0 1.5])
title('Training Data for Classifier 2 (3 artists, same genre)',...
    'FontSize',14)
legend('Artist 1 - AC/DC','Artist 2 - Aerosmith',...
    'Artist 3 - The Rolling Stones','threshold 1','threshold 2')
print(gcf,'Test2_dataplot.png','-dpng')
%% Test 3 - Country vs Classical vs Rock
clear;clc;

[m1,Fs] = audioread('Mozart1.wav');
[Mozart(:,2),Fs] = audioread('Mozart2.wav');
[Mozart(:,3),Fs] = audioread('Mozart3.wav');
[Mozart(:,4),Fs] = audioread('Mozart4.wav');
[Mozart(:,5),Fs] = audioread('Mozart5.wav');
[Mozart(:,6),Fs] = audioread('Mozart6.wav');
[m7,Fs] = audioread('Mozart7.wav');
[Mozart(:,8),Fs] = audioread('Mozart8.wav');
[Mozart(:,9),Fs] = audioread('Mozart9.wav');
```

```
[Mozart(:,10),Fs] = audioread('Mozart10.wav');
[m11,Fs] = audioread('Mozart11.wav');
[Mozart(:,12),Fs] = audioread('Mozart12.wav');
[m13,Fs] = audioread('Mozart13.wav');
[m14,Fs] = audioread('Mozart14.wav');
[m15,Fs] = audioread('Mozart15.wav');
[Mozart(:,16),Fs] = audioread('Mozart16.wav');
[Mozart(:,17),Fs] = audioread('Mozart17.wav');
[Mozart(:,18),Fs] = audioread('Mozart18.wav');
[Mozart(:,19),Fs] = audioread('Mozart19.wav');
[Mozart(:,20),Fs] = audioread('Mozart20.wav');
[m21,Fs] = audioread('Mozart21.wav');
[Mozart(:,22),Fs] = audioread('Mozart22.wav');
[Mozart(:,23),Fs] = audioread('Mozart23.wav');
[Mozart(:,24),Fs] = audioread('Mozart24.wav');
[Mozart(:,25),Fs] = audioread('Mozart25.wav');
[Mozart(:,26),Fs] = audioread('Mozart26.wav');
[Mozart(:,27),Fs] = audioread('Mozart27.wav');
[Mozart(:,28),Fs] = audioread('Mozart28.wav');
[Mozart(:,29),Fs] = audioread('Mozart29.wav');
[Mozart(:,30),Fs] = audioread('Mozart30.wav');
Mozart = Mozart(1:220059,:);
Mozart(:,1) = m1(1:220059);
Mozart(:,7) = m7(1:220059);
Mozart(:,11) = m11(1:220059);
Mozart(:,13) = m13(1:220059);
Mozart(:,14) = m14(1:220059);
Mozart(:,15) = m15(1:220059);
Mozart(:,21) = m21(1:220059);

[y1,Fs] = audioread('YoYoMa1.wav');
[YoYoMa(:,2),Fs] = audioread('YoYoMa2.wav');
[y3,Fs] = audioread('YoYoMa3.wav');
[YoYoMa(:,4),Fs] = audioread('YoYoMa4.wav');
[YoYoMa(:,5),Fs] = audioread('YoYoMa5.wav');
[YoYoMa(:,6),Fs] = audioread('YoYoMa6.wav');
[y8,Fs] = audioread('YoYoMa8.wav');
[YoYoMa(:,9),Fs] = audioread('YoYoMa9.wav');
[YoYoMa(:,10),Fs] = audioread('YoYoMa10.wav');
[y11,Fs] = audioread('YoYoMa11.wav');
[YoYoMa(:,12),Fs] = audioread('YoYoMa12.wav');
[YoYoMa(:,13),Fs] = audioread('YoYoMa13.wav');
[YoYoMa(:,14),Fs] = audioread('YoYoMa14.wav');
[YoYoMa(:,15),Fs] = audioread('YoYoMa15.wav');
[YoYoMa(:,16),Fs] = audioread('YoYoMa16.wav');
[y17,Fs] = audioread('YoYoMa17.wav');
```

```
[YoYoMa(:,18),Fs] = audioread('YoYoMa18.wav');
[YoYoMa(:,19),Fs] = audioread('YoYoMa19.wav');
[YoYoMa(:,20),Fs] = audioread('YoYoMa20.wav');
[y21,Fs] = audioread('YoYoMa21.wav');
[y22,Fs] = audioread('YoYoMa22.wav');
[y23,Fs] = audioread('YoYoMa23.wav');
[YoYoMa(:,24),Fs] = audioread('YoYoMa24.wav');
[YoYoMa(:,25),Fs] = audioread('YoYoMa25.wav');
[y26,Fs] = audioread('YoYoMa26.wav');
[YoYoMa(:,27),Fs] = audioread('YoYoMa27.wav');
[YoYoMa(:,28),Fs] = audioread('YoYoMa28.wav');
[YoYoMa(:,29),Fs] = audioread('YoYoMa29.wav');
[YoYoMa(:,30),Fs] = audioread('YoYoMa30.wav');
YoYoMa = YoYoMa(1:219547,:);
YoYoMa(:,1) = y1(1:219547);
YoYoMa(:,3) = y3(1:219547);
YoYoMa(:,8) = y8(1:219547);
YoYoMa(:,11) = y11(1:219547);
YoYoMa(:,17) = y17(1:219547);
YoYoMa(:,21) = y21(1:219547);
YoYoMa(:,22) = y22(1:219547);
YoYoMa(:,23) = y23(1:219547);
YoYoMa(:,26) = y26(1:219547);

[Chesney(:,1),Fs] = audioread('Chesney1.wav');
[Chesney(:,2),Fs] = audioread('Chesney2.wav');
[Chesney(:,3),Fs] = audioread('Chesney3.wav');
[Chesney(:,4),Fs] = audioread('Chesney4.wav');
[Chesney(:,5),Fs] = audioread('Chesney5.wav');
[Chesney(:,6),Fs] = audioread('Chesney6.wav');
[Chesney(:,7),Fs] = audioread('Chesney7.wav');
[Chesney(:,8),Fs] = audioread('Chesney8.wav');
[c9,Fs] = audioread('Chesney9.wav');
[Chesney(:,10),Fs] = audioread('Chesney10.wav');
[Chesney(:,11),Fs] = audioread('Chesney11.wav');
[Chesney(:,12),Fs] = audioread('Chesney12.wav');
[Chesney(:,13),Fs] = audioread('Chesney13.wav');
[Chesney(:,14),Fs] = audioread('Chesney14.wav');
[Chesney(:,15),Fs] = audioread('Chesney15.wav');
[Chesney(:,16),Fs] = audioread('Chesney16.wav');
[Chesney(:,17),Fs] = audioread('Chesney17.wav');
[c18,Fs] = audioread('Chesney18.wav');
[c19,Fs] = audioread('Chesney19.wav');
[Chesney(:,20),Fs] = audioread('Chesney20.wav');
[Chesney(:,21),Fs] = audioread('Chesney21.wav');
[Chesney(:,22),Fs] = audioread('Chesney22.wav');
```

```
[Chesney(:,23),Fs] = audioread('Chesney23.wav');
[Chesney(:,24),Fs] = audioread('Chesney24.wav');
[Chesney(:,25),Fs] = audioread('Chesney25.wav');
[Chesney(:,26),Fs] = audioread('Chesney26.wav');
[Chesney(:,27),Fs] = audioread('Chesney27.wav');
[Chesney(:,28),Fs] = audioread('Chesney28.wav');
[Chesney(:,29),Fs] = audioread('Chesney29.wav');
[Chesney(:,30),Fs] = audioread('Chesney30.wav');
Chesney = Chesney(1:220672,:);
Chesney(:,9) = c9(1:220672);
Chesney(:,18) = c18(1:220672);
Chesney(:,19) = c19(1:220672);

[Bentley(:,1),Fs] = audioread('Bentley1.wav');
[Bentley(:,2),Fs] = audioread('Bentley2.wav');
[b3,Fs] = audioread('Bentley3.wav');
[Bentley(:,4),Fs] = audioread('Bentley4.wav');
[Bentley(:,5),Fs] = audioread('Bentley5.wav');
[Bentley(:,6),Fs] = audioread('Bentley6.wav');
[b7,Fs] = audioread('Bentley7.wav');
[Bentley(:,8),Fs] = audioread('Bentley8.wav');
[Bentley(:,9),Fs] = audioread('Bentley9.wav');
[Bentley(:,10),Fs] = audioread('Bentley10.wav');
[Bentley(:,11),Fs] = audioread('Bentley11.wav');
[Bentley(:,12),Fs] = audioread('Bentley12.wav');
[Bentley(:,13),Fs] = audioread('Bentley13.wav');
[Bentley(:,14),Fs] = audioread('Bentley14.wav');
[b15,Fs] = audioread('Bentley15.wav');
[Bentley(:,16),Fs] = audioread('Bentley16.wav');
[Bentley(:,17),Fs] = audioread('Bentley17.wav');
[Bentley(:,18),Fs] = audioread('Bentley18.wav');
[Bentley(:,19),Fs] = audioread('Bentley19.wav');
[Bentley(:,20),Fs] = audioread('Bentley20.wav');
[Bentley(:,21),Fs] = audioread('Bentley21.wav');
[Bentley(:,22),Fs] = audioread('Bentley22.wav');
[Bentley(:,23),Fs] = audioread('Bentley23.wav');
[b24,Fs] = audioread('Bentley24.wav');
[Bentley(:,25),Fs] = audioread('Bentley25.wav');
[Bentley(:,26),Fs] = audioread('Bentley26.wav');
[Bentley(:,27),Fs] = audioread('Bentley27.wav');
[Bentley(:,28),Fs] = audioread('Bentley28.wav');
[Bentley(:,29),Fs] = audioread('Bentley29.wav');
[b30,Fs] = audioread('Bentley30.wav');
Bentley = Bentley(1:221184,:);
Bentley(:,3) = b3(1:221184);
Bentley(:,7) = b7(1:221184);
Bentley(:,25) = b15(1:221184);
Bentley(:,24) = b24(1:221184);
Bentley(:,30) = b30(1:221184);

[Combs(:,1),Fs] = audioread('Combs1.wav');
[Combs(:,2),Fs] = audioread('Combs2.wav');
[Combs(:,3),Fs] = audioread('Combs3.wav');
[Combs(:,4),Fs] = audioread('Combs4.wav');
[x5,Fs] = audioread('Combs5.wav');
[Combs(:,6),Fs] = audioread('Combs6.wav');
[x7,Fs] = audioread('Combs7.wav');
[x8,Fs] = audioread('Combs8.wav');
[x9,Fs] = audioread('Combs9.wav');
[x10,Fs] = audioread('Combs10.wav');
[x11,Fs] = audioread('Combs11.wav');
[Combs(:,12),Fs] = audioread('Combs12.wav');
[x13,Fs] = audioread('Combs13.wav');
[Combs(:,14),Fs] = audioread('Combs14.wav');
[Combs(:,15),Fs] = audioread('Combs15.wav');
[Combs(:,16),Fs] = audioread('Combs16.wav');
[Combs(:,17),Fs] = audioread('Combs17.wav');
[Combs(:,18),Fs] = audioread('Combs18.wav');
[Combs(:,19),Fs] = audioread('Combs19.wav');
[Combs(:,20),Fs] = audioread('Combs20.wav');
[Combs(:,21),Fs] = audioread('Combs21.wav');
[x22,Fs] = audioread('Combs22.wav');
[Combs(:,23),Fs] = audioread('Combs23.wav');
[Combs(:,24),Fs] = audioread('Combs24.wav');
[Combs(:,25),Fs] = audioread('Combs25.wav');
[Combs(:,26),Fs] = audioread('Combs26.wav');
[Combs(:,27),Fs] = audioread('Combs27.wav');
[Combs(:,28),Fs] = audioread('Combs28.wav');
[Combs(:,29),Fs] = audioread('Combs29.wav');
[Combs(:,30),Fs] = audioread('Combs30.wav');
Combs = Combs(1:220160,:);
Combs(:,5) = x5(1:220160);
Combs(:,7) = x7(1:220160);
Combs(:,8) = x8(1:220160);
Combs(:,9) = x9(1:220160);
Combs(:,10) = x10(1:220160);
Combs(:,11) = x11(1:220160);
Combs(:,13) = x13(1:220160);
Combs(:,22) = x22(1:220160);

[Bach(:,1),Fs] = audioread('Bach1.wav');
[Bach(:,2),Fs] = audioread('Bach2.wav');
```

```
[Bach(:,3),Fs] = audioread('Bach3.wav');
[x4,Fs] = audioread('Bach4.wav');
[Bach(:,5),Fs] = audioread('Bach5.wav');
[x6,Fs] = audioread('Bach6.wav');
[Bach(:,7),Fs] = audioread('Bach7.wav');
[x8,Fs] = audioread('Bach8.wav');
[Bach(:,9),Fs] = audioread('Bach9.wav');
[x10,Fs] = audioread('Bach10.wav');
[Bach(:,11),Fs] = audioread('Bach11.wav');
[x12,Fs] = audioread('Bach12.wav');
[Bach(:,13),Fs] = audioread('Bach13.wav');
[x14,Fs] = audioread('Bach14.wav');
[Bach(:,15),Fs] = audioread('Bach15.wav');
[Bach(:,16),Fs] = audioread('Bach16.wav');
[Bach(:,17),Fs] = audioread('Bach17.wav');
[Bach(:,18),Fs] = audioread('Bach18.wav');
[Bach(:,19),Fs] = audioread('Bach19.wav');
[Bach(:,20),Fs] = audioread('Bach20.wav');
[Bach(:,21),Fs] = audioread('Bach21.wav');
[Bach(:,22),Fs] = audioread('Bach22.wav');
[Bach(:,23),Fs] = audioread('Bach23.wav');
[x24,Fs] = audioread('Bach24.wav');
[Bach(:,25),Fs] = audioread('Bach25.wav');
[Bach(:,26),Fs] = audioread('Bach26.wav');
[Bach(:,27),Fs] = audioread('Bach27.wav');
[Bach(:,28),Fs] = audioread('Bach28.wav');
[Bach(:,29),Fs] = audioread('Bach29.wav');
[Bach(:,30),Fs] = audioread('Bach30.wav');
Bach = Bach(1:221184,:);
Bach(:,4) = x4(1:221184);
Bach(:,6) = x6(1:221184);
Bach(:,8) = x8(1:221184);
Bach(:,10) = x10(1:221184);
Bach(:,12) = x12(1:221184);
Bach(:,14) = x14(1:221184);
Bach(:,24) = x24(1:221184);

[x1,Fs] = audioread('Stones1.wav');
[Stones(:,2),Fs] = audioread('Stones2.wav');
[Stones(:,3),Fs] = audioread('Stones3.wav');
[Stones(:,4),Fs] = audioread('Stones4.wav');
[Stones(:,5),Fs] = audioread('Stones5.wav');
[Stones(:,6),Fs] = audioread('Stones6.wav');
[Stones(:,7),Fs] = audioread('Stones7.wav');
[x8,Fs] = audioread('Stones8.wav');
[Stones(:,9),Fs] = audioread('Stones9.wav');
```

```
[Stones(:,10),Fs] = audioread('Stones10.wav');
[x11,Fs] = audioread('Stones11.wav');
[Stones(:,12),Fs] = audioread('Stones12.wav');
[x13,Fs] = audioread('Stones13.wav');
[Stones(:,14),Fs] = audioread('Stones14.wav');
[Stones(:,15),Fs] = audioread('Stones15.wav');
[Stones(:,16),Fs] = audioread('Stones16.wav');
[x17,Fs] = audioread('Stones17.wav');
[x18,Fs] = audioread('Stones18.wav');
[x19,Fs] = audioread('Stones19.wav');
[x20,Fs] = audioread('Stones20.wav');
[Stones(:,21),Fs] = audioread('Stones21.wav');
[x22,Fs] = audioread('Stones22.wav');
[Stones(:,23),Fs] = audioread('Stones23.wav');
[Stones(:,24),Fs] = audioread('Stones24.wav');
[Stones(:,25),Fs] = audioread('Stones25.wav');
[x26,Fs] = audioread('Stones26.wav');
[x27,Fs] = audioread('Stones27.wav');
[x28,Fs] = audioread('Stones28.wav');
[x29,Fs] = audioread('Stones29.wav');
[x30,Fs] = audioread('Stones30.wav');
Stones = Stones(1:220672,:);
Stones(:,1) = x1(1:220672);
Stones(:,8) = x8(1:220672);
Stones(:,11) = x11(1:220672);
Stones(:,13) = x13(1:220672);
Stones(:,17) = x17(1:220672);
Stones(:,18) = x18(1:220672);
Stones(:,19) = x19(1:220672);
Stones(:,20) = x20(1:220672);
Stones(:,22) = x22(1:220672);
Stones(:,26) = x26(1:220672);
Stones(:,27) = x27(1:220672);
Stones(:,28) = x28(1:220672);
Stones(:,29) = x29(1:220672);
Stones(:,30) = x30(1:220672);

[ACDC(:,1),Fs] = audioread('ACDC1.wav');
[ACDC(:,2),Fs] = audioread('ACDC2.wav');
[ACDC(:,3),Fs] = audioread('ACDC3.wav');
[ACDC(:,4),Fs] = audioread('ACDC4.wav');
[ACDC(:,5),Fs] = audioread('ACDC5.wav');
[ACDC(:,6),Fs] = audioread('ACDC6.wav');
[ACDC(:,7),Fs] = audioread('ACDC7.wav');
[z8,Fs] = audioread('ACDC8.wav');
[z9,Fs] = audioread('ACDC9.wav');
```

```
[z10,Fs] = audioread('ACDC10.wav');
[ACDC(:,11),Fs] = audioread('ACDC11.wav');
[z12,Fs] = audioread('ACDC12.wav');
[z13,Fs] = audioread('ACDC13.wav');
[ACDC(:,14),Fs] = audioread('ACDC14.wav');
[ACDC(:,15),Fs] = audioread('ACDC15.wav');
[ACDC(:,16),Fs] = audioread('ACDC16.wav');
[z17,Fs] = audioread('ACDC17.wav');
[ACDC(:,18),Fs] = audioread('ACDC18.wav');
[ACDC(:,19),Fs] = audioread('ACDC19.wav');
[ACDC(:,20),Fs] = audioread('ACDC20.wav');
[z21,Fs] = audioread('ACDC21.wav');
[ACDC(:,22),Fs] = audioread('ACDC22.wav');
[ACDC(:,23),Fs] = audioread('ACDC23.wav');
[z24,Fs] = audioread('ACDC24.wav');
[ACDC(:,25),Fs] = audioread('ACDC25.wav');
[ACDC(:,26),Fs] = audioread('ACDC26.wav');
[ACDC(:,27),Fs] = audioread('ACDC27.wav');
[z28,Fs] = audioread('ACDC28.wav');
[ACDC(:,29),Fs] = audioread('ACDC29.wav');
[ACDC(:,30),Fs] = audioread('ACDC30.wav');
ACDC = ACDC(1:220571,:);
ACDC(:,8) = z8(1:220571);
ACDC(:,9) = z9(1:220571);
ACDC(:,10) = z10(1:220571);
ACDC(:,12) = z12(1:220571);
ACDC(:,13) = z13(1:220571);
ACDC(:,17) = z17(1:220571);
ACDC(:,21) = z21(1:220571);
ACDC(:,24) = z24(1:220571);
ACDC(:,28) = z28(1:220571);

[AS(:,1),Fs] = audioread('Aerosmith1.wav');
[AS(:,2),Fs] = audioread('Aerosmith2.wav');
[AS(:,3),Fs] = audioread('Aerosmith3.wav');
[AS(:,4),Fs] = audioread('Aerosmith4.wav');
[AS(:,5),Fs] = audioread('Aerosmith5.wav');
[AS(:,6),Fs] = audioread('Aerosmith6.wav');
[AS(:,7),Fs] = audioread('Aerosmith7.wav');
[AS(:,8),Fs] = audioread('Aerosmith8.wav');
[AS(:,9),Fs] = audioread('Aerosmith9.wav');
[AS(:,10),Fs] = audioread('Aerosmith10.wav');
[AS(:,11),Fs] = audioread('Aerosmith11.wav');
[AS(:,12),Fs] = audioread('Aerosmith12.wav');
[AS(:,13),Fs] = audioread('Aerosmith13.wav');
[AS(:,14),Fs] = audioread('Aerosmith14.wav');
```

```
[AS(:,15),Fs] = audioread('Aerosmith15.wav');
[AS(:,16),Fs] = audioread('Aerosmith16.wav');
[AS(:,17),Fs] = audioread('Aerosmith17.wav');
[w18,Fs] = audioread('Aerosmith18.wav');
[AS(:,19),Fs] = audioread('Aerosmith19.wav');
[AS(:,20),Fs] = audioread('Aerosmith20.wav');
[AS(:,21),Fs] = audioread('Aerosmith21.wav');
[AS(:,22),Fs] = audioread('Aerosmith22.wav');
[AS(:,23),Fs] = audioread('Aerosmith23.wav');
[AS(:,24),Fs] = audioread('Aerosmith24.wav');
[AS(:,25),Fs] = audioread('Aerosmith25.wav');
[AS(:,26),Fs] = audioread('Aerosmith26.wav');
[AS(:,27),Fs] = audioread('Aerosmith27.wav');
[AS(:,28),Fs] = audioread('Aerosmith28.wav');
[AS(:,29),Fs] = audioread('Aerosmith29.wav');
[AS(:,30),Fs] = audioread('Aerosmith30.wav');
AS = AS(1:220160,:);
AS(:,18) = w18;


ACDC = ACDC(1:219547,:);
AS = AS(1:219547,:);
Stones = Stones(1:219547,:);
rock = [ACDC AS Stones];


Bentley = Bentley(1:219547,:);
Chesney = Chesney(1:219547,:);
Combs = Combs(1:219547,:);
country = [Bentley Chesney Combs];

Bach = Bach(1:219547,:);
Mozart = Mozart(1:219547,:);
YoYoMa = YoYoMa(1:219547,:);
classical = [Bach Mozart YoYoMa];
feature = 20;

Specclassical = zeros(262152,90);
Speccountry = zeros(262152,90);
Specrock = zeros(262152,90);
for jj = 1:90
    S1 = spectrogram(classical(:,jj));
    S2 = spectrogram(country(:,jj));
    S3 = spectrogram(rock(:,jj));
    Specclassical(:,jj) = abs(S1(:));
    Speccountry(:,jj) = abs(S2(:));
    Specrock(:,jj) = abs(S3(:));
end


[U,S,V,threshold1,threshold2,w,sortgenre1,sortgenre2,sortgenre3] =...
    test3_trainer(Specclassical,Speccountry,Specrock,feature);

[Test(:,1),Fs] = audioread('ACDCTest1.wav');
[Test(:,2),Fs] = audioread('ACDCTest2.wav');
[Test(:,3),Fs] = audioread('ACDCTest3.wav');
[Test(:,4),Fs] = audioread('ACDCTest4.wav');
[Test(:,5),Fs] = audioread('ACDCTest5.wav');
[Test(:,6),Fs] = audioread('ACDCTest6.wav');
[Test(:,7),Fs] = audioread('ACDCTest7.wav');
[Test(:,8),Fs] = audioread('ASTest1.wav');
[Test(:,9),Fs] = audioread('ASTest2.wav');
[Test(:,10),Fs] = audioread('ASTest3.wav');
[Test(:,11),Fs] = audioread('ASTest4.wav');
[Test(:,12),Fs] = audioread('ASTest5.wav');
[Test(:,13),Fs] = audioread('ASTest6.wav');
[Test(:,14),Fs] = audioread('ASTest7.wav');
[Test15,Fs] = audioread('StonesTest1.wav');
[Test16,Fs] = audioread('StonesTest2.wav');
[Test17,Fs] = audioread('StonesTest3.wav');
[Test(:,18),Fs] = audioread('StonesTest4.wav');
[Test19,Fs] = audioread('StonesTest5.wav');
[Test(:,20),Fs] = audioread('StonesTest6.wav');
[Test(:,21),Fs] = audioread('StonesTest7.wav');
TestR = Test(1:220160,:);
TestR(:,15) = Test15(1:220160);
TestR(:,16) = Test16(1:220160);
TestR(:,17) = Test17(1:220160);
TestR(:,19) = Test19(1:220160);

[Test(:,1),Fs] = audioread('BachTest7.wav');
[Test2,Fs] = audioread('BachTest2.wav');
[Test3,Fs] = audioread('BachTest3.wav');
[Test4,Fs] = audioread('BachTest4.wav');
[Test5,Fs] = audioread('BachTest5.wav');
[Test6,Fs] = audioread('BachTest6.wav');
[Test7,Fs] = audioread('BachTest1.wav');
[Test(:,8),Fs] = audioread('MozartTest1.wav');
[Test(:,9),Fs] = audioread('MozartTest2.wav');
[Test(:,10),Fs] = audioread('MozartTest3.wav');
[Test(:,11),Fs] = audioread('MozartTest4.wav');
[Test(:,12),Fs] = audioread('MozartTest5.wav');
```

```
[Test(:,13),Fs] = audioread('MozartTest6.wav');
[Test(:,14),Fs] = audioread('MozartTest7.wav');
[Test(:,15),Fs] = audioread('YoYoMaTest1.wav');
[Test(:,16),Fs] = audioread('YoYoMaTest2.wav');
[Test(:,17),Fs] = audioread('YoYoMaTest3.wav');
[Test(:,18),Fs] = audioread('YoYoMaTest4.wav');
[Test(:,19),Fs] = audioread('YoYoMaTest5.wav');
[Test(:,20),Fs] = audioread('YoYoMaTest6.wav');
[Test(:,21),Fs] = audioread('YoYoMaTest7.wav');
TestClas = Test(1:220160,:);
TestClas(:,2) = Test2(1:220160);
TestClas(:,3) = Test3(1:220160);
TestClas(:,4) = Test4(1:220160);
TestClas(:,5) = Test5(1:220160);
TestClas(:,6) = Test6(1:220160);
TestClas(:,7) = Test7(1:220160);


[Test(:,1),Fs] = audioread('ChesneyTest1.wav');
[Test(:,2),Fs] = audioread('ChesneyTest2.wav');
[Test(:,3),Fs] = audioread('ChesneyTest3.wav');
[Test(:,4),Fs] = audioread('ChesneyTest4.wav');
[Test(:,5),Fs] = audioread('ChesneyTest5.wav');
[Test(:,6),Fs] = audioread('ChesneyTest6.wav');
[Test(:,7),Fs] = audioread('ChesneyTest7.wav');
[Test8,Fs] = audioread('CombsTest1.wav');
[Test9,Fs] = audioread('CombsTest2.wav');
[Test(:,10),Fs] = audioread('CombsTest3.wav');
[Test11,Fs] = audioread('CombsTest4.wav');
[Test12,Fs] = audioread('CombsTest5.wav');
[Test13,Fs] = audioread('CombsTest6.wav');
[Test14,Fs] = audioread('CombsTest7.wav');
[Test(:,15),Fs] = audioread('BentleyTest1.wav');
[Test(:,16),Fs] = audioread('BentleyTest2.wav');
[Test(:,17),Fs] = audioread('BentleyTest3.wav');
[Test(:,18),Fs] = audioread('BentleyTest4.wav');
[Test(:,19),Fs] = audioread('BentleyTest5.wav');
[Test(:,20),Fs] = audioread('BentleyTest6.wav');
[Test(:,21),Fs] = audioread('BentleyTest7.wav');
TestCount = Test(1:220160,:);
TestCount(:,8) = Test8(1:220160);
TestCount(:,9) = Test9(1:220160);
TestCount(:,11) = Test11(1:220160);
TestCount(:,12) = Test12(1:220160);
TestCount(:,13) = Test13(1:220160);
TestCount(:,14) = Test14(1:220160);
```

```

TestGenre = [TestClas TestCount TestR];

SpecTest = zeros(262152,63);
for jj = 1:63
    S1 = spectrogram(TestGenre(:,jj));
    SpecTest(:,jj) = abs(S1(:));
end
%%
TestMatrix = U'*SpecTest;
projection = w'*TestMatrix;
for j = 1:21
    True(:,j) = [1;0];
True(:,j+21) = [0;0];
True(:,j+42) = [1;1];
end
ResVec(1,:) = (projection>threshold1);
ResVec(2,:) = (projection>threshold2)
True-ResVec
%% Plot classical/country/rock projections onto w
figure(3)
p = plot(sortgenre1,zeros(90),'ob','Linewidth',2)
for j = 2:90
set(get(get(p(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
hold on
p1 = plot(sortgenre2,.5*ones(90),'dr','Linewidth',2)
for j = 2:90
set(get(get(p1(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
p2 = plot(sortgenre3,ones(90),'*g','Linewidth',2)
for j = 2:90
set(get(get(p2(j),'Annotation'),'LegendInformation'),...
    'IconDisplayStyle','off');
end
plot([threshold1 threshold1],[0 10],'r')
plot([threshold2 threshold2],[0 10],'r')
ylim([0 1.5])
title('Training Data for Classifier 3 (3 genres, 3 artists in each)',...
    'FontSize',14)
legend('Genre 1 - Classical','Genre 2 - Country',...
    'Genre 3 - Rock','threshold 1','threshold 2')
print(gcf,'Test3_dataplot.png','-dpng')
%% Functions
```

```
function [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] ...
= test1_trainer(artist1,artist2,artist3,feature)
size_art1 = size(artist1,2);
size_art2 = size(artist2,2);
size_art3 = size(artist3,2);
[U,S,V] = svd([artist1 artist2 artist3],'econ');
artists = S*V'; % projection onto principal components
U = U(:,1:feature);
artist1songs = artists(1:feature,1:size_art1);
artist2songs = artists(1:feature,size_art1+1:size_art1+size_art2);
artist3songs = artists(1:feature,size_art1+size_art2+1:size_art1+...
    size_art2+size_art3);
mean_art1 = mean(artist1songs,2);
mean_art2 = mean(artist2songs,2);
mean_art3 = mean(artist3songs,2);
Sw = 0; % within class variances
for k=1:size_art1
    Sw = Sw + (artist1songs(:,k)-mean_art1)*(artist1songs(:,k)-mean_art1)';
end
for k=1:size_art2
    Sw = Sw + (artist2songs(:,k)-mean_art2)*(artist2songs(:,k)-mean_art2)';
end
for k=1:size_art3
    Sw = Sw + (artist3songs(:,k)-mean_art3)*(artist3songs(:,k)-mean_art3)';
end

sample_mean = (mean_art1+mean_art2+mean_art3)/3;
Sb = 30*(mean_art1-sample_mean)*(mean_art1-sample_mean)';
Sb = Sb + 30*(mean_art2-sample_mean)*(mean_art2-sample_mean)';
Sb = Sb + 30*(mean_art3-sample_mean)*(mean_art3-sample_mean)'; % between class
[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);
v_art1 = w'*artist1songs;
v_art2 = w'*artist2songs;
v_art3 = w'*artist3songs;
%art1 on left and art3 on right
if mean(v_art3)>mean(v_art2)
    w = -w;
    v_art2 = -v_art2;
    v_art3 = -v_art3;
end
%art1 < threshold1 < art2 < threshold2 < art3
    sortart1 = sort(v_art1);
    sortart2 = sort(v_art2);
    sortart3 = sort(v_art3);
    t1 = length(sortart1);
    t2 = 1;
    while sortart1(t1)>sortart3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sortart1(t1)+sortart3(t2))/2;

    t1 = length(sortart2);
    t2 = 1;
    while sortart3(t1)>sortart2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sortart3(t1)+sortart2(t2))/2;
end

function [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
    test2_trainer(artist1,artist2,artist3,feature)
size_art1 = size(artist1,2);
size_art2 = size(artist2,2);
size_art3 = size(artist3,2);
[U,S,V] = svd([artist1 artist2 artist3],'econ');
artists = S*V'; % projection onto principal components
U = U(:,1:feature);
artist1songs = abs(artists(1:feature,1:size_art1));
artist2songs = abs(artists(1:feature,size_art1+1:size_art1+size_art2));
artist3songs = abs(artists...
    (1:feature,size_art1+size_art2+1:size_art1+size_art2+size_art3));
mean_art1 = mean(artist1songs,2);
mean_art2 = mean(artist2songs,2);
mean_art3 = mean(artist3songs,2);
Sw = 0; % within class variances
for k=1:size_art1
    Sw = Sw + (artist1songs(:,k)-mean_art1)*(artist1songs(:,k)-mean_art1)';
end
for k=1:size_art2
    Sw = Sw + (artist2songs(:,k)-mean_art2)*(artist2songs(:,k)-mean_art2)';
end
for k=1:size_art3
    Sw = Sw + (artist3songs(:,k)-mean_art3)*(artist3songs(:,k)-mean_art3)';
end

sample_mean = (mean_art1+mean_art2+mean_art3)/3;
Sb = 30*(mean_art1-sample_mean)*(mean_art1-sample_mean)';
Sb = Sb + 30*(mean_art2-sample_mean)*(mean_art2-sample_mean)';
Sb = Sb + 30*(mean_art3-sample_mean)*(mean_art3-sample_mean)'; % between class
[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);
v_art1 = w'*artist1songs;
v_art2 = w'*artist2songs;
v_art3 = w'*artist3songs;
%art1 on left and art3 on right
if mean(v_art2)>mean(v_art1)
    w = -w;
    v_art2 = -v_art2;
    v_art1 = -v_art1;
end
%art1 < threshold1 < art2 < threshold2 < art3
    sortart1 = sort(v_art1);
    sortart2 = sort(v_art2);
    sortart3 = sort(v_art3);
    t1 = length(sortart1);
    t2 = 1;
    while sortart1(t1)>sortart2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sortart1(t1)+sortart2(t2))/2;

    t1 = length(sortart2);
    t2 = 1;
    while sortart1(t1)>sortart3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sortart1(t1)+sortart3(t2))/2;
end


function [U,S,V,threshold1,threshold2,w,sortgenre1,sortgenre2,sortgenre3] =...
    test3_trainer(genre1,genre2,genre3,feature)
size_genre1 = size(genre1,2);
size_genre2 = size(genre2,2);
size_genre3 = size(genre3,2);
[U,S,V] = svd([genre1 genre2 genre3],'econ');
genres = S*V'; % projection onto principal components
U = U(:,1:feature);
genre1songs = genres(1:feature,1:size_genre1);
genre2songs = genres(1:feature,size_genre1+1:size_genre1+size_genre2);
genre3songs = genres(1:feature,size_genre1+size_genre2+1:size_genre1+...
    size_genre2+size_genre3);
mean_genre1 = mean(genre1songs,2);
mean_genre2 = mean(genre2songs,2);
mean_genre3 = mean(genre3songs,2);
Sw = 0; % within class variances
for k=1:size_genre1
    Sw = Sw + (genre1songs(:,k)-mean_genre1)*(genre1songs(:,k)-mean_genre1)';
end
for k=1:size_genre2
    Sw = Sw + (genre2songs(:,k)-mean_genre2)*(genre2songs(:,k)-mean_genre2)';
end
for k=1:size_genre3
    Sw = Sw + (genre3songs(:,k)-mean_genre3)*(genre3songs(:,k)-mean_genre3)';
end

sample_mean = (mean_genre1+mean_genre2+mean_genre3)/3;
Sb = 90*(mean_genre1-sample_mean)*(mean_genre1-sample_mean)';
Sb = Sb + 90*(mean_genre2-sample_mean)*(mean_genre2-sample_mean)';
Sb = Sb + 90*(mean_genre3-sample_mean)*...
    (mean_genre3-sample_mean)'; % between class
[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);
v_genre1 = w'*genre1songs;
v_genre2 = w'*genre2songs;
v_genre3 = w'*genre3songs;
%art1 on left and art3 on right
if mean(v_genre1)>mean(v_genre3)
    w = -w;
    v_genre3 = -v_genre3;
    v_genre1 = -v_genre1;
end
%art1 < threshold1 < art2 < threshold2 < art3
    sortgenre1 = sort(v_genre1);
    sortgenre2 = sort(v_genre2);
    t1 = length(sortgenre1);
    t2 = 1;
    while sortgenre2(t1)>sortgenre1(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sortgenre2(t1)+sortgenre1(t2))/2;
    %threshold1 = -710.165;

    sortgenre3 = sort(v_genre3)
    t2 = length(sortgenre2);
    t3 = 1;
    while sortgenre1(t2)>sortgenre3(t3)
        t2 = t2-1
        t3 = t3+1
    end
    threshold2 = abs((sortgenre1(t2)+sortgenre3(t3))/2);
    %threshold2 = 1692.165;
end
```