

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:

$$A = \hat{U} \hat{\Sigma} V^T \quad (1)$$

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:

$$w = \arg \max_w \frac{w^T S_B w}{w^T S_W w} \quad (2)$$

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

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

$$S_W = \sum_{j=1}^n \sum_{\vec{x}} (\vec{x} - \vec{\mu}_j)(\vec{x} - \vec{\mu}_j)^T \quad (4)$$

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:

$$S_B w = \lambda S_W w \quad (5)$$

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 begin 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 spectrograms) 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 min-

imizes 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

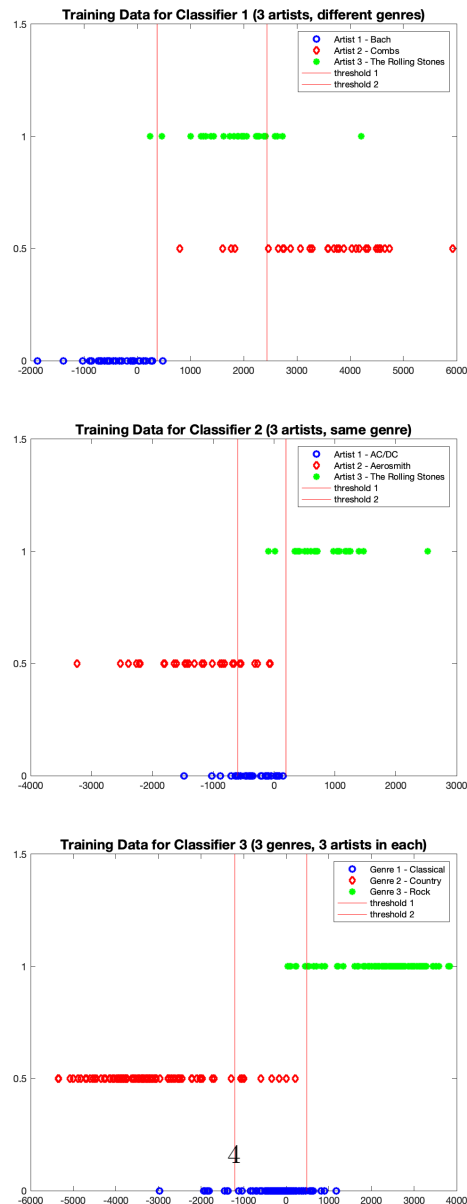


Figure 1: Above are the projected data and statistically decided thresholds for each classifier.

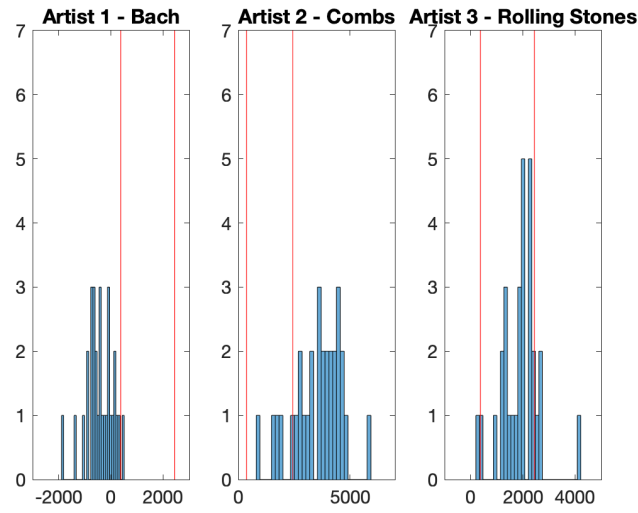


Figure 2: Above shows the amount of song data that was captured in each section of the Classifier 1.

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

```

1 %% Test 1 – Luke Combs vs Bach vs Rolling Stones
2 clear; clc; close all;
3
4 [Combs(:,1),Fs] = audioread('Combs1.wav');
5 [Combs(:,2),Fs] = audioread('Combs2.wav');
6 [Combs(:,3),Fs] = audioread('Combs3.wav');
7 [Combs(:,4),Fs] = audioread('Combs4.wav');
8 [x5,Fs] = audioread('Combs5.wav');
```

```

9  [Combs(:,6),Fs] = audioread('Combs6.wav');
10 [x7,Fs] = audioread('Combs7.wav');
11 [x8,Fs] = audioread('Combs8.wav');
12 [x9,Fs] = audioread('Combs9.wav');
13 [x10,Fs] = audioread('Combs10.wav');
14 [x11,Fs] = audioread('Combs11.wav');
15 [Combs(:,12),Fs] = audioread('Combs12.wav');
16 [x13,Fs] = audioread('Combs13.wav');
17 [Combs(:,14),Fs] = audioread('Combs14.wav');
18 [Combs(:,15),Fs] = audioread('Combs15.wav');
19 [Combs(:,16),Fs] = audioread('Combs16.wav');
20 [Combs(:,17),Fs] = audioread('Combs17.wav');
21 [Combs(:,18),Fs] = audioread('Combs18.wav');
22 [Combs(:,19),Fs] = audioread('Combs19.wav');
23 [Combs(:,20),Fs] = audioread('Combs20.wav');
24 [Combs(:,21),Fs] = audioread('Combs21.wav');
25 [x22,Fs] = audioread('Combs22.wav');
26 [Combs(:,23),Fs] = audioread('Combs23.wav');
27 [Combs(:,24),Fs] = audioread('Combs24.wav');
28 [Combs(:,25),Fs] = audioread('Combs25.wav');
29 [Combs(:,26),Fs] = audioread('Combs26.wav');
30 [Combs(:,27),Fs] = audioread('Combs27.wav');
31 [Combs(:,28),Fs] = audioread('Combs28.wav');
32 [Combs(:,29),Fs] = audioread('Combs29.wav');
33 [Combs(:,30),Fs] = audioread('Combs30.wav');
34 Combs = Combs(1:220160,:);
35 Combs(:,5) = x5(1:220160);
36 Combs(:,7) = x7(1:220160);
37 Combs(:,8) = x8(1:220160);
38 Combs(:,9) = x9(1:220160);
39 Combs(:,10) = x10(1:220160);
40 Combs(:,11) = x11(1:220160);
41 Combs(:,13) = x13(1:220160);
42 Combs(:,22) = x22(1:220160);
43
44 [Bach(:,1),Fs] = audioread('Bach1.wav');
45 [Bach(:,2),Fs] = audioread('Bach2.wav');
46 [Bach(:,3),Fs] = audioread('Bach3.wav');
47 [x4,Fs] = audioread('Bach4.wav');
48 [Bach(:,5),Fs] = audioread('Bach5.wav');
49 [x6,Fs] = audioread('Bach6.wav');
50 [Bach(:,7),Fs] = audioread('Bach7.wav');
51 [x8,Fs] = audioread('Bach8.wav');
52 [Bach(:,9),Fs] = audioread('Bach9.wav');
53 [x10,Fs] = audioread('Bach10.wav');
54 [Bach(:,11),Fs] = audioread('Bach11.wav');

```

```

55 [x12,Fs] = audioread('Bach12.wav');
56 [Bach(:,13),Fs] = audioread('Bach13.wav');
57 [x14,Fs] = audioread('Bach14.wav');
58 [Bach(:,15),Fs] = audioread('Bach15.wav');
59 [Bach(:,16),Fs] = audioread('Bach16.wav');
60 [Bach(:,17),Fs] = audioread('Bach17.wav');
61 [Bach(:,18),Fs] = audioread('Bach18.wav');
62 [Bach(:,19),Fs] = audioread('Bach19.wav');
63 [Bach(:,20),Fs] = audioread('Bach20.wav');
64 [Bach(:,21),Fs] = audioread('Bach21.wav');
65 [Bach(:,22),Fs] = audioread('Bach22.wav');
66 [Bach(:,23),Fs] = audioread('Bach23.wav');
67 [x24,Fs] = audioread('Bach24.wav');
68 [Bach(:,25),Fs] = audioread('Bach25.wav');
69 [Bach(:,26),Fs] = audioread('Bach26.wav');
70 [Bach(:,27),Fs] = audioread('Bach27.wav');
71 [Bach(:,28),Fs] = audioread('Bach28.wav');
72 [Bach(:,29),Fs] = audioread('Bach29.wav');
73 [Bach(:,30),Fs] = audioread('Bach30.wav');
74 Bach = Bach(1:221184,:);
75 Bach(:,4) = x4(1:221184);
76 Bach(:,6) = x6(1:221184);
77 Bach(:,8) = x8(1:221184);
78 Bach(:,10) = x10(1:221184);
79 Bach(:,12) = x12(1:221184);
80 Bach(:,14) = x14(1:221184);
81 Bach(:,24) = x24(1:221184);
82
83 [x1,Fs] = audioread('Stones1.wav');
84 [Stones(:,2),Fs] = audioread('Stones2.wav');
85 [Stones(:,3),Fs] = audioread('Stones3.wav');
86 [Stones(:,4),Fs] = audioread('Stones4.wav');
87 [Stones(:,5),Fs] = audioread('Stones5.wav');
88 [Stones(:,6),Fs] = audioread('Stones6.wav');
89 [Stones(:,7),Fs] = audioread('Stones7.wav');
90 [x8,Fs] = audioread('Stones8.wav');
91 [Stones(:,9),Fs] = audioread('Stones9.wav');
92 [Stones(:,10),Fs] = audioread('Stones10.wav');
93 [x11,Fs] = audioread('Stones11.wav');
94 [Stones(:,12),Fs] = audioread('Stones12.wav');
95 [x13,Fs] = audioread('Stones13.wav');
96 [Stones(:,14),Fs] = audioread('Stones14.wav');
97 [Stones(:,15),Fs] = audioread('Stones15.wav');
98 [Stones(:,16),Fs] = audioread('Stones16.wav');
99 [x17,Fs] = audioread('Stones17.wav');
100 [x18,Fs] = audioread('Stones18.wav');

```

```

101 [x19,Fs] = audioread('Stones19.wav');
102 [x20,Fs] = audioread('Stones20.wav');
103 [Stones(:,21),Fs] = audioread('Stones21.wav');
104 [x22,Fs] = audioread('Stones22.wav');
105 [Stones(:,23),Fs] = audioread('Stones23.wav');
106 [Stones(:,24),Fs] = audioread('Stones24.wav');
107 [Stones(:,25),Fs] = audioread('Stones25.wav');
108 [x26,Fs] = audioread('Stones26.wav');
109 [x27,Fs] = audioread('Stones27.wav');
110 [x28,Fs] = audioread('Stones28.wav');
111 [x29,Fs] = audioread('Stones29.wav');
112 [x30,Fs] = audioread('Stones30.wav');
113 Stones = Stones(1:220672,:);
114 Stones(:,1) = x1(1:220672);
115 Stones(:,8) = x8(1:220672);
116 Stones(:,11) = x11(1:220672);
117 Stones(:,13) = x13(1:220672);
118 Stones(:,17) = x17(1:220672);
119 Stones(:,18) = x18(1:220672);
120 Stones(:,19) = x19(1:220672);
121 Stones(:,20) = x20(1:220672);
122 Stones(:,22) = x22(1:220672);
123 Stones(:,26) = x26(1:220672);
124 Stones(:,27) = x27(1:220672);
125 Stones(:,28) = x28(1:220672);
126 Stones(:,29) = x29(1:220672);
127 Stones(:,30) = x30(1:220672);
128
129 Stones = Stones(1:220160,:);
130 Bach = Bach(1:220160,:);
131 feature = 15;
132
133 SpecBach = zeros(262152,30);
134 SpecCombs = zeros(262152,30);
135 SpecStones = zeros(262152,30);
136 for jj = 1:30
137     S1 = spectrogram(Bach(:,jj));
138     S2 = spectrogram(Combs(:,jj));
139     S3 = spectrogram(Stones(:,jj));
140     SpecBach(:,jj) = abs(S1(:));
141     SpecCombs(:,jj) = abs(S2(:));
142     SpecStones(:,jj) = abs(S3(:));
143 end
144
145 [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
146 test1_trainer(SpecBach,SpecCombs,SpecStones,feature);

```



```

147
148 [Test(:,1),Fs] = audioread('BachTest1.wav');
149 [Test(:,2),Fs] = audioread('BachTest2.wav');
150 [Test3,Fs] = audioread('BachTest3.wav');
151 [Test4,Fs] = audioread('BachTest4.wav');
152 [Test5,Fs] = audioread('BachTest5.wav');
153 [Test6,Fs] = audioread('BachTest6.wav');
154 [Test7,Fs] = audioread('BachTest7.wav');
155 [Test(:,8),Fs] = audioread('CombsTest1.wav');
156 [Test(:,9),Fs] = audioread('CombsTest2.wav');
157 [Test10,Fs] = audioread('CombsTest3.wav');
158 [Test11,Fs] = audioread('CombsTest4.wav');
159 [Test12,Fs] = audioread('CombsTest5.wav');
160 [Test13,Fs] = audioread('CombsTest6.wav');
161 [Test14,Fs] = audioread('CombsTest7.wav');
162 [Test15,Fs] = audioread('StonesTest1.wav');
163 [Test16,Fs] = audioread('StonesTest2.wav');
164 [Test17,Fs] = audioread('StonesTest3.wav');
165 [Test18,Fs] = audioread('StonesTest4.wav');
166 [Test19,Fs] = audioread('StonesTest5.wav');
167 [Test20,Fs] = audioread('StonesTest6.wav');
168 [Test21,Fs] = audioread('StonesTest7.wav');
169 Test = Test(1:220160,:);
170 Test(:,3) = Test3(1:220160);
171 Test(:,4) = Test4(1:220160);
172 Test(:,5) = Test5(1:220160);
173 Test(:,6) = Test6(1:220160);
174 Test(:,7) = Test7(1:220160);
175 Test(:,10) = Test10(1:220160);
176 Test(:,11) = Test11(1:220160);
177 Test(:,12) = Test12(1:220160);
178 Test(:,13) = Test13(1:220160);
179 Test(:,14) = Test14(1:220160);
180 Test(:,15) = Test15(1:220160);
181 Test(:,16) = Test16(1:220160);
182 Test(:,17) = Test17(1:220160);
183 Test(:,18) = Test18(1:220160);
184 Test(:,19) = Test19(1:220160);
185 Test(:,20) = Test20(1:220160);
186 Test(:,21) = Test21(1:220160);
187
188 SpecTest = zeros(262152,21);
189 for jj = 1:21
190     S1 = spectrogram(Test(:,jj));
191     SpecTest(:,jj) = abs(S1(:));
192 end

```

```

193 %%
194 TestMatrix = U*SpecTest;
195 projection = w*TestMatrix;
196 True = [0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1;
197         0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0];
198 ResVec(1,:) = (projection>threshold1);
199 ResVec(2,:) = (projection>threshold2)
200 True-ResVec
201 %% Plot artist1/2/3 projections onto w
202 figure(1)
203 p = plot(sortart1,zeros(30),'ob','Linewidth',2)
204 for j = 2:30
205     set(get(get(p(j),'Annotation'),'LegendInformation'),...
206         'IconDisplayStyle','off');
207 end
208 hold on
209 p1 = plot(sortart2,.5*ones(30),'dr','Linewidth',2)
210 for j = 2:30
211     set(get(get(p1(j),'Annotation'),'LegendInformation'),...
212         'IconDisplayStyle','off');
213 end
214 p2 = plot(sortart3,ones(30),'*g','Linewidth',2)
215 for j = 2:30
216     set(get(get(p2(j),'Annotation'),'LegendInformation'),...
217         'IconDisplayStyle','off');
218 end
219 plot([threshold1 threshold1],[0 10],'r')
220 plot([threshold2 threshold2],[0 10],'r')
221 ylim([0 1.5])
222 title('Training Data for Classifier 1 (3 artists, different genres)')...
223     , 'FontSize',14)
224 legend('Artist 1 - Bach','Artist 2 - Combs',...
225         'Artist 3 - The Rolling Stones','threshold 1','threshold 2')
226 print(gcf,'Test1_dataplot.png','-dpng')
227 %% Plot histogram of training data with thresholds
228 figure(5)
229 subplot(1,3,1)
230 histogram(sortart1,30);
231 hold on, plot([threshold1 threshold1],[0 7],'r')
232 plot([threshold2 threshold2],[0 7],'r')
233 set(gca,'Xlim',[-3000 3000],'Ylim',[0 7],'FontSize',14)
234 title('Artist 1 - Bach')
235 subplot(1,3,2)
236 histogram(sortart2,30);
237 hold on,
238 plot([threshold1 threshold1],[0 7],'r')

```

```

239 plot([threshold2 threshold2],[0 7], 'r')
240 set(gca, 'Xlim',[0 7000], 'Ylim',[0 7], 'FontSize',14)
241 title('Artist 2 - Combs')
242 subplot(1,3,3)
243 histogram(sortart3,30);
244 hold on, plot([threshold1 threshold1],[0 7], 'r')
245 plot([threshold2 threshold2],[0 7], 'r')
246 set(gca, 'Xlim',[-1000 5000], 'Ylim',[0 7], 'FontSize',14)
247 title('Artist 3 - Rolling Stones')
248 print(gcf, 'Test1_histogram.png', '-dpng')
249 %% Test 2 - AC/DC vs Rolling Stones vs Aerosmith
250 clear; clc;
251 [ACDC(:,1),Fs] = audioread('ACDC1.wav');
252 [ACDC(:,2),Fs] = audioread('ACDC2.wav');
253 [ACDC(:,3),Fs] = audioread('ACDC3.wav');
254 [ACDC(:,4),Fs] = audioread('ACDC4.wav');
255 [ACDC(:,5),Fs] = audioread('ACDC5.wav');
256 [ACDC(:,6),Fs] = audioread('ACDC6.wav');
257 [ACDC(:,7),Fs] = audioread('ACDC7.wav');
258 [z8,Fs] = audioread('ACDC8.wav');
259 [z9,Fs] = audioread('ACDC9.wav');
260 [z10,Fs] = audioread('ACDC10.wav');
261 [ACDC(:,11),Fs] = audioread('ACDC11.wav');
262 [z12,Fs] = audioread('ACDC12.wav');
263 [z13,Fs] = audioread('ACDC13.wav');
264 [ACDC(:,14),Fs] = audioread('ACDC14.wav');
265 [ACDC(:,15),Fs] = audioread('ACDC15.wav');
266 [ACDC(:,16),Fs] = audioread('ACDC16.wav');
267 [z17,Fs] = audioread('ACDC17.wav');
268 [ACDC(:,18),Fs] = audioread('ACDC18.wav');
269 [ACDC(:,19),Fs] = audioread('ACDC19.wav');
270 [ACDC(:,20),Fs] = audioread('ACDC20.wav');
271 [z21,Fs] = audioread('ACDC21.wav');
272 [ACDC(:,22),Fs] = audioread('ACDC22.wav');
273 [ACDC(:,23),Fs] = audioread('ACDC23.wav');
274 [z24,Fs] = audioread('ACDC24.wav');
275 [ACDC(:,25),Fs] = audioread('ACDC25.wav');
276 [ACDC(:,26),Fs] = audioread('ACDC26.wav');
277 [ACDC(:,27),Fs] = audioread('ACDC27.wav');
278 [z28,Fs] = audioread('ACDC28.wav');
279 [ACDC(:,29),Fs] = audioread('ACDC29.wav');
280 [ACDC(:,30),Fs] = audioread('ACDC30.wav');
281 ACDC = ACDC(1:220571,:);
282 ACDC(:,8) = z8(1:220571);
283 ACDC(:,9) = z9(1:220571);
284 ACDC(:,10) = z10(1:220571);

```

```

285 ACDC(:,12) = z12(1:220571);
286 ACDC(:,13) = z13(1:220571);
287 ACDC(:,17) = z17(1:220571);
288 ACDC(:,21) = z21(1:220571);
289 ACDC(:,24) = z24(1:220571);
290 ACDC(:,28) = z28(1:220571);
291
292 [AS(:,1),Fs] = audioread('Aerosmith1.wav');
293 [AS(:,2),Fs] = audioread('Aerosmith2.wav');
294 [AS(:,3),Fs] = audioread('Aerosmith3.wav');
295 [AS(:,4),Fs] = audioread('Aerosmith4.wav');
296 [AS(:,5),Fs] = audioread('Aerosmith5.wav');
297 [AS(:,6),Fs] = audioread('Aerosmith6.wav');
298 [AS(:,7),Fs] = audioread('Aerosmith7.wav');
299 [AS(:,8),Fs] = audioread('Aerosmith8.wav');
300 [AS(:,9),Fs] = audioread('Aerosmith9.wav');
301 [AS(:,10),Fs] = audioread('Aerosmith10.wav');
302 [AS(:,11),Fs] = audioread('Aerosmith11.wav');
303 [AS(:,12),Fs] = audioread('Aerosmith12.wav');
304 [AS(:,13),Fs] = audioread('Aerosmith13.wav');
305 [AS(:,14),Fs] = audioread('Aerosmith14.wav');
306 [AS(:,15),Fs] = audioread('Aerosmith15.wav');
307 [AS(:,16),Fs] = audioread('Aerosmith16.wav');
308 [AS(:,17),Fs] = audioread('Aerosmith17.wav');
309 [w18,Fs] = audioread('Aerosmith18.wav');
310 [AS(:,19),Fs] = audioread('Aerosmith19.wav');
311 [AS(:,20),Fs] = audioread('Aerosmith20.wav');
312 [AS(:,21),Fs] = audioread('Aerosmith21.wav');
313 [AS(:,22),Fs] = audioread('Aerosmith22.wav');
314 [AS(:,23),Fs] = audioread('Aerosmith23.wav');
315 [AS(:,24),Fs] = audioread('Aerosmith24.wav');
316 [AS(:,25),Fs] = audioread('Aerosmith25.wav');
317 [AS(:,26),Fs] = audioread('Aerosmith26.wav');
318 [AS(:,27),Fs] = audioread('Aerosmith27.wav');
319 [AS(:,28),Fs] = audioread('Aerosmith28.wav');
320 [AS(:,29),Fs] = audioread('Aerosmith29.wav');
321 [AS(:,30),Fs] = audioread('Aerosmith30.wav');
322 AS = AS(1:220160,:);
323 AS(:,18) = w18;
324
325 [x1,Fs] = audioread('Stones1.wav');
326 [Stones(:,2),Fs] = audioread('Stones2.wav');
327 [Stones(:,3),Fs] = audioread('Stones3.wav');
328 [Stones(:,4),Fs] = audioread('Stones4.wav');
329 [Stones(:,5),Fs] = audioread('Stones5.wav');
330 [Stones(:,6),Fs] = audioread('Stones6.wav');

```

```

331 [Stones(:,7),Fs] = audioread('Stones7.wav');
332 [x8,Fs] = audioread('Stones8.wav');
333 [Stones(:,9),Fs] = audioread('Stones9.wav');
334 [Stones(:,10),Fs] = audioread('Stones10.wav');
335 [x11,Fs] = audioread('Stones11.wav');
336 [Stones(:,12),Fs] = audioread('Stones12.wav');
337 [x13,Fs] = audioread('Stones13.wav');
338 [Stones(:,14),Fs] = audioread('Stones14.wav');
339 [Stones(:,15),Fs] = audioread('Stones15.wav');
340 [Stones(:,16),Fs] = audioread('Stones16.wav');
341 [x17,Fs] = audioread('Stones17.wav');
342 [x18,Fs] = audioread('Stones18.wav');
343 [x19,Fs] = audioread('Stones19.wav');
344 [x20,Fs] = audioread('Stones20.wav');
345 [Stones(:,21),Fs] = audioread('Stones21.wav');
346 [x22,Fs] = audioread('Stones22.wav');
347 [Stones(:,23),Fs] = audioread('Stones23.wav');
348 [Stones(:,24),Fs] = audioread('Stones24.wav');
349 [Stones(:,25),Fs] = audioread('Stones25.wav');
350 [x26,Fs] = audioread('Stones26.wav');
351 [x27,Fs] = audioread('Stones27.wav');
352 [x28,Fs] = audioread('Stones28.wav');
353 [x29,Fs] = audioread('Stones29.wav');
354 [x30,Fs] = audioread('Stones30.wav');
355 Stones = Stones(1:220672,:);
356 Stones(:,1) = x1(1:220672);
357 Stones(:,8) = x8(1:220672);
358 Stones(:,11) = x11(1:220672);
359 Stones(:,13) = x13(1:220672);
360 Stones(:,17) = x17(1:220672);
361 Stones(:,18) = x18(1:220672);
362 Stones(:,19) = x19(1:220672);
363 Stones(:,20) = x20(1:220672);
364 Stones(:,22) = x22(1:220672);
365 Stones(:,26) = x26(1:220672);
366 Stones(:,27) = x27(1:220672);
367 Stones(:,28) = x28(1:220672);
368 Stones(:,29) = x29(1:220672);
369 Stones(:,30) = x30(1:220672);
370
371 Stones = Stones(1:220160,:);
372 ACDC = ACDC(1:220160,:);
373 feature = 20;
374
375 SpecACDC = zeros(262152,30);
376 SpecAS = zeros(262152,30);

```

```

377 SpecStones = zeros(262152,30);
378 for jj = 1:30
379     S1 = spectrogram(ACDC(:,jj));
380     S2 = spectrogram(AS(:,jj));
381     S3 = spectrogram(Stones(:,jj));
382     SpecACDC(:,jj) = abs(S1(:));
383     SpecAS(:,jj) = abs(S2(:));
384     SpecStones(:,jj) = abs(S3(:));
385 end
386
387 [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
388 test2_trainer(SpecACDC,SpecAS,SpecStones,feature);
389
390 [Test(:,1),Fs] = audioread('ACDCTest1.wav');
391 [Test(:,2),Fs] = audioread('ACDCTest2.wav');
392 [Test(:,3),Fs] = audioread('ACDCTest3.wav');
393 [Test(:,4),Fs] = audioread('ACDCTest4.wav');
394 [Test(:,5),Fs] = audioread('ACDCTest5.wav');
395 [Test(:,6),Fs] = audioread('ACDCTest6.wav');
396 [Test(:,7),Fs] = audioread('ACDCTest7.wav');
397 [Test(:,8),Fs] = audioread('ASTest1.wav');
398 [Test(:,9),Fs] = audioread('ASTest2.wav');
399 [Test(:,10),Fs] = audioread('ASTest3.wav');
400 [Test(:,11),Fs] = audioread('ASTest4.wav');
401 [Test(:,12),Fs] = audioread('ASTest5.wav');
402 [Test(:,13),Fs] = audioread('ASTest6.wav');
403 [Test(:,14),Fs] = audioread('ASTest7.wav');
404 [Test15,Fs] = audioread('StonesTest1.wav');
405 [Test16,Fs] = audioread('StonesTest2.wav');
406 [Test17,Fs] = audioread('StonesTest3.wav');
407 [Test(:,18),Fs] = audioread('StonesTest4.wav');
408 [Test19,Fs] = audioread('StonesTest5.wav');
409 [Test(:,20),Fs] = audioread('StonesTest6.wav');
410 [Test(:,21),Fs] = audioread('StonesTest7.wav');
411
412 Test = Test(1:220160,:);
413 Test(:,15) = Test15(1:220160);
414 Test(:,16) = Test16(1:220160);
415 Test(:,17) = Test17(1:220160);
416 Test(:,19) = Test19(1:220160);
417
418 SpecTest = zeros(262152,21);
419 for jj = 1:21
420     S1 = spectrogram(Test(:,jj));
421     SpecTest(:,jj) = abs(S1(:));
422 end

```

```

423 %%
424 TestMatrix = U'*SpecTest;
425 projection = w'*TestMatrix;
426 True = [1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1;
427         0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1];
428 ResVec(1,:) = (projection>threshold1);
429 ResVec(2,:) = (projection>threshold2)
430 True-ResVec
431 %% Plot artist1/2/3 projections onto w
432 figure(2)
433 p = plot(sortart1,zeros(30),'ob','Linewidth',2)
434 for j = 2:30
435     set(get(get(p(j),'Annotation'),'LegendInformation'),...
436         'IconDisplayStyle','off');
437 end
438 hold on
439 p1 = plot(sortart2,.5*ones(30),'dr','Linewidth',2)
440 for j = 2:30
441     set(get(get(p1(j),'Annotation'),'LegendInformation'),...
442         'IconDisplayStyle','off');
443 end
444 p2 = plot(sortart3,ones(30),'*g','Linewidth',2)
445 for j = 2:30
446     set(get(get(p2(j),'Annotation'),'LegendInformation'),...
447         'IconDisplayStyle','off');
448 end
449 plot([threshold1 threshold1],[0 10],'r')
450 plot([threshold2 threshold2],[0 10],'r')
451 ylim([0 1.5])
452 title('Training Data for Classifier 2 (3 artists, same genre)',...
453       'FontSize',14)
454 legend('Artist 1 - AC/DC','Artist 2 - Aerosmith',...
455        'Artist 3 - The Rolling Stones','threshold 1','threshold 2')
456 print(gcf,'Test2_dataplot.png','-dpng')
457 %% Test 3 - Country vs Classical vs Rock
458 clear;clc;
459
460 [m1,Fs] = audioread('Mozart1.wav');
461 [Mozart(:,2),Fs] = audioread('Mozart2.wav');
462 [Mozart(:,3),Fs] = audioread('Mozart3.wav');
463 [Mozart(:,4),Fs] = audioread('Mozart4.wav');
464 [Mozart(:,5),Fs] = audioread('Mozart5.wav');
465 [Mozart(:,6),Fs] = audioread('Mozart6.wav');
466 [m7,Fs] = audioread('Mozart7.wav');
467 [Mozart(:,8),Fs] = audioread('Mozart8.wav');
468 [Mozart(:,9),Fs] = audioread('Mozart9.wav');

```

```

469 [Mozart(:,10),Fs] = audioread('Mozart10.wav');
470 [m11,Fs] = audioread('Mozart11.wav');
471 [Mozart(:,12),Fs] = audioread('Mozart12.wav');
472 [m13,Fs] = audioread('Mozart13.wav');
473 [m14,Fs] = audioread('Mozart14.wav');
474 [m15,Fs] = audioread('Mozart15.wav');
475 [Mozart(:,16),Fs] = audioread('Mozart16.wav');
476 [Mozart(:,17),Fs] = audioread('Mozart17.wav');
477 [Mozart(:,18),Fs] = audioread('Mozart18.wav');
478 [Mozart(:,19),Fs] = audioread('Mozart19.wav');
479 [Mozart(:,20),Fs] = audioread('Mozart20.wav');
480 [m21,Fs] = audioread('Mozart21.wav');
481 [Mozart(:,22),Fs] = audioread('Mozart22.wav');
482 [Mozart(:,23),Fs] = audioread('Mozart23.wav');
483 [Mozart(:,24),Fs] = audioread('Mozart24.wav');
484 [Mozart(:,25),Fs] = audioread('Mozart25.wav');
485 [Mozart(:,26),Fs] = audioread('Mozart26.wav');
486 [Mozart(:,27),Fs] = audioread('Mozart27.wav');
487 [Mozart(:,28),Fs] = audioread('Mozart28.wav');
488 [Mozart(:,29),Fs] = audioread('Mozart29.wav');
489 [Mozart(:,30),Fs] = audioread('Mozart30.wav');
490 Mozart = Mozart(1:220059,:);
491 Mozart(:,1) = m1(1:220059);
492 Mozart(:,7) = m7(1:220059);
493 Mozart(:,11) = m11(1:220059);
494 Mozart(:,13) = m13(1:220059);
495 Mozart(:,14) = m14(1:220059);
496 Mozart(:,15) = m15(1:220059);
497 Mozart(:,21) = m21(1:220059);
498
499 [y1,Fs] = audioread('YoYoMa1.wav');
500 [YoYoMa(:,2),Fs] = audioread('YoYoMa2.wav');
501 [y3,Fs] = audioread('YoYoMa3.wav');
502 [YoYoMa(:,4),Fs] = audioread('YoYoMa4.wav');
503 [YoYoMa(:,5),Fs] = audioread('YoYoMa5.wav');
504 [YoYoMa(:,6),Fs] = audioread('YoYoMa6.wav');
505 [y8,Fs] = audioread('YoYoMa8.wav');
506 [YoYoMa(:,9),Fs] = audioread('YoYoMa9.wav');
507 [YoYoMa(:,10),Fs] = audioread('YoYoMa10.wav');
508 [y11,Fs] = audioread('YoYoMa11.wav');
509 [YoYoMa(:,12),Fs] = audioread('YoYoMa12.wav');
510 [YoYoMa(:,13),Fs] = audioread('YoYoMa13.wav');
511 [YoYoMa(:,14),Fs] = audioread('YoYoMa14.wav');
512 [YoYoMa(:,15),Fs] = audioread('YoYoMa15.wav');
513 [YoYoMa(:,16),Fs] = audioread('YoYoMa16.wav');
514 [y17,Fs] = audioread('YoYoMa17.wav');

```



```

515 [YoYoMa(:,18),Fs] = audioread('YoYoMa18.wav');
516 [YoYoMa(:,19),Fs] = audioread('YoYoMa19.wav');
517 [YoYoMa(:,20),Fs] = audioread('YoYoMa20.wav');
518 [y21,Fs] = audioread('YoYoMa21.wav');
519 [y22,Fs] = audioread('YoYoMa22.wav');
520 [y23,Fs] = audioread('YoYoMa23.wav');
521 [YoYoMa(:,24),Fs] = audioread('YoYoMa24.wav');
522 [YoYoMa(:,25),Fs] = audioread('YoYoMa25.wav');
523 [y26,Fs] = audioread('YoYoMa26.wav');
524 [YoYoMa(:,27),Fs] = audioread('YoYoMa27.wav');
525 [YoYoMa(:,28),Fs] = audioread('YoYoMa28.wav');
526 [YoYoMa(:,29),Fs] = audioread('YoYoMa29.wav');
527 [YoYoMa(:,30),Fs] = audioread('YoYoMa30.wav');
528 YoYoMa = YoYoMa(1:219547,:);
529 YoYoMa(:,1) = y1(1:219547);
530 YoYoMa(:,3) = y3(1:219547);
531 YoYoMa(:,8) = y8(1:219547);
532 YoYoMa(:,11) = y11(1:219547);
533 YoYoMa(:,17) = y17(1:219547);
534 YoYoMa(:,21) = y21(1:219547);
535 YoYoMa(:,22) = y22(1:219547);
536 YoYoMa(:,23) = y23(1:219547);
537 YoYoMa(:,26) = y26(1:219547);
538
539 [Chesney(:,1),Fs] = audioread('Chesney1.wav');
540 [Chesney(:,2),Fs] = audioread('Chesney2.wav');
541 [Chesney(:,3),Fs] = audioread('Chesney3.wav');
542 [Chesney(:,4),Fs] = audioread('Chesney4.wav');
543 [Chesney(:,5),Fs] = audioread('Chesney5.wav');
544 [Chesney(:,6),Fs] = audioread('Chesney6.wav');
545 [Chesney(:,7),Fs] = audioread('Chesney7.wav');
546 [Chesney(:,8),Fs] = audioread('Chesney8.wav');
547 [c9,Fs] = audioread('Chesney9.wav');
548 [Chesney(:,10),Fs] = audioread('Chesney10.wav');
549 [Chesney(:,11),Fs] = audioread('Chesney11.wav');
550 [Chesney(:,12),Fs] = audioread('Chesney12.wav');
551 [Chesney(:,13),Fs] = audioread('Chesney13.wav');
552 [Chesney(:,14),Fs] = audioread('Chesney14.wav');
553 [Chesney(:,15),Fs] = audioread('Chesney15.wav');
554 [Chesney(:,16),Fs] = audioread('Chesney16.wav');
555 [Chesney(:,17),Fs] = audioread('Chesney17.wav');
556 [c18,Fs] = audioread('Chesney18.wav');
557 [c19,Fs] = audioread('Chesney19.wav');
558 [Chesney(:,20),Fs] = audioread('Chesney20.wav');
559 [Chesney(:,21),Fs] = audioread('Chesney21.wav');
560 [Chesney(:,22),Fs] = audioread('Chesney22.wav');

```

```

561 [Chesney(:,23),Fs] = audioread('Chesney23.wav');
562 [Chesney(:,24),Fs] = audioread('Chesney24.wav');
563 [Chesney(:,25),Fs] = audioread('Chesney25.wav');
564 [Chesney(:,26),Fs] = audioread('Chesney26.wav');
565 [Chesney(:,27),Fs] = audioread('Chesney27.wav');
566 [Chesney(:,28),Fs] = audioread('Chesney28.wav');
567 [Chesney(:,29),Fs] = audioread('Chesney29.wav');
568 [Chesney(:,30),Fs] = audioread('Chesney30.wav');
569 Chesney = Chesney(1:220672,:);
570 Chesney(:,9) = c9(1:220672);
571 Chesney(:,18) = c18(1:220672);
572 Chesney(:,19) = c19(1:220672);
573
574 [Bentley(:,1),Fs] = audioread('Bentley1.wav');
575 [Bentley(:,2),Fs] = audioread('Bentley2.wav');
576 [b3,Fs] = audioread('Bentley3.wav');
577 [Bentley(:,4),Fs] = audioread('Bentley4.wav');
578 [Bentley(:,5),Fs] = audioread('Bentley5.wav');
579 [Bentley(:,6),Fs] = audioread('Bentley6.wav');
580 [b7,Fs] = audioread('Bentley7.wav');
581 [Bentley(:,8),Fs] = audioread('Bentley8.wav');
582 [Bentley(:,9),Fs] = audioread('Bentley9.wav');
583 [Bentley(:,10),Fs] = audioread('Bentley10.wav');
584 [Bentley(:,11),Fs] = audioread('Bentley11.wav');
585 [Bentley(:,12),Fs] = audioread('Bentley12.wav');
586 [Bentley(:,13),Fs] = audioread('Bentley13.wav');
587 [Bentley(:,14),Fs] = audioread('Bentley14.wav');
588 [b15,Fs] = audioread('Bentley15.wav');
589 [Bentley(:,16),Fs] = audioread('Bentley16.wav');
590 [Bentley(:,17),Fs] = audioread('Bentley17.wav');
591 [Bentley(:,18),Fs] = audioread('Bentley18.wav');
592 [Bentley(:,19),Fs] = audioread('Bentley19.wav');
593 [Bentley(:,20),Fs] = audioread('Bentley20.wav');
594 [Bentley(:,21),Fs] = audioread('Bentley21.wav');
595 [Bentley(:,22),Fs] = audioread('Bentley22.wav');
596 [Bentley(:,23),Fs] = audioread('Bentley23.wav');
597 [b24,Fs] = audioread('Bentley24.wav');
598 [Bentley(:,25),Fs] = audioread('Bentley25.wav');
599 [Bentley(:,26),Fs] = audioread('Bentley26.wav');
600 [Bentley(:,27),Fs] = audioread('Bentley27.wav');
601 [Bentley(:,28),Fs] = audioread('Bentley28.wav');
602 [Bentley(:,29),Fs] = audioread('Bentley29.wav');
603 [b30,Fs] = audioread('Bentley30.wav');
604 Bentley = Bentley(1:221184,:);
605 Bentley(:,3) = b3(1:221184);
606 Bentley(:,7) = b7(1:221184);

```

```

607 Bentley(:,25) = b15(1:221184);
608 Bentley(:,24) = b24(1:221184);
609 Bentley(:,30) = b30(1:221184);
610
611 [Combs(:,1),Fs] = audioread('Combs1.wav');
612 [Combs(:,2),Fs] = audioread('Combs2.wav');
613 [Combs(:,3),Fs] = audioread('Combs3.wav');
614 [Combs(:,4),Fs] = audioread('Combs4.wav');
615 [x5,Fs] = audioread('Combs5.wav');
616 [Combs(:,6),Fs] = audioread('Combs6.wav');
617 [x7,Fs] = audioread('Combs7.wav');
618 [x8,Fs] = audioread('Combs8.wav');
619 [x9,Fs] = audioread('Combs9.wav');
620 [x10,Fs] = audioread('Combs10.wav');
621 [x11,Fs] = audioread('Combs11.wav');
622 [Combs(:,12),Fs] = audioread('Combs12.wav');
623 [x13,Fs] = audioread('Combs13.wav');
624 [Combs(:,14),Fs] = audioread('Combs14.wav');
625 [Combs(:,15),Fs] = audioread('Combs15.wav');
626 [Combs(:,16),Fs] = audioread('Combs16.wav');
627 [Combs(:,17),Fs] = audioread('Combs17.wav');
628 [Combs(:,18),Fs] = audioread('Combs18.wav');
629 [Combs(:,19),Fs] = audioread('Combs19.wav');
630 [Combs(:,20),Fs] = audioread('Combs20.wav');
631 [Combs(:,21),Fs] = audioread('Combs21.wav');
632 [x22,Fs] = audioread('Combs22.wav');
633 [Combs(:,23),Fs] = audioread('Combs23.wav');
634 [Combs(:,24),Fs] = audioread('Combs24.wav');
635 [Combs(:,25),Fs] = audioread('Combs25.wav');
636 [Combs(:,26),Fs] = audioread('Combs26.wav');
637 [Combs(:,27),Fs] = audioread('Combs27.wav');
638 [Combs(:,28),Fs] = audioread('Combs28.wav');
639 [Combs(:,29),Fs] = audioread('Combs29.wav');
640 [Combs(:,30),Fs] = audioread('Combs30.wav');
641 Combs = Combs(1:220160,:);
642 Combs(:,5) = x5(1:220160);
643 Combs(:,7) = x7(1:220160);
644 Combs(:,8) = x8(1:220160);
645 Combs(:,9) = x9(1:220160);
646 Combs(:,10) = x10(1:220160);
647 Combs(:,11) = x11(1:220160);
648 Combs(:,13) = x13(1:220160);
649 Combs(:,22) = x22(1:220160);
650
651 [Bach(:,1),Fs] = audioread('Bach1.wav');
652 [Bach(:,2),Fs] = audioread('Bach2.wav');

```

```

653 [Bach(:,3),Fs] = audioread('Bach3.wav');
654 [x4,Fs] = audioread('Bach4.wav');
655 [Bach(:,5),Fs] = audioread('Bach5.wav');
656 [x6,Fs] = audioread('Bach6.wav');
657 [Bach(:,7),Fs] = audioread('Bach7.wav');
658 [x8,Fs] = audioread('Bach8.wav');
659 [Bach(:,9),Fs] = audioread('Bach9.wav');
660 [x10,Fs] = audioread('Bach10.wav');
661 [Bach(:,11),Fs] = audioread('Bach11.wav');
662 [x12,Fs] = audioread('Bach12.wav');
663 [Bach(:,13),Fs] = audioread('Bach13.wav');
664 [x14,Fs] = audioread('Bach14.wav');
665 [Bach(:,15),Fs] = audioread('Bach15.wav');
666 [Bach(:,16),Fs] = audioread('Bach16.wav');
667 [Bach(:,17),Fs] = audioread('Bach17.wav');
668 [Bach(:,18),Fs] = audioread('Bach18.wav');
669 [Bach(:,19),Fs] = audioread('Bach19.wav');
670 [Bach(:,20),Fs] = audioread('Bach20.wav');
671 [Bach(:,21),Fs] = audioread('Bach21.wav');
672 [Bach(:,22),Fs] = audioread('Bach22.wav');
673 [Bach(:,23),Fs] = audioread('Bach23.wav');
674 [x24,Fs] = audioread('Bach24.wav');
675 [Bach(:,25),Fs] = audioread('Bach25.wav');
676 [Bach(:,26),Fs] = audioread('Bach26.wav');
677 [Bach(:,27),Fs] = audioread('Bach27.wav');
678 [Bach(:,28),Fs] = audioread('Bach28.wav');
679 [Bach(:,29),Fs] = audioread('Bach29.wav');
680 [Bach(:,30),Fs] = audioread('Bach30.wav');
681 Bach = Bach(1:221184,:);
682 Bach(:,4) = x4(1:221184);
683 Bach(:,6) = x6(1:221184);
684 Bach(:,8) = x8(1:221184);
685 Bach(:,10) = x10(1:221184);
686 Bach(:,12) = x12(1:221184);
687 Bach(:,14) = x14(1:221184);
688 Bach(:,24) = x24(1:221184);
689
690 [x1,Fs] = audioread('Stones1.wav');
691 [Stones(:,2),Fs] = audioread('Stones2.wav');
692 [Stones(:,3),Fs] = audioread('Stones3.wav');
693 [Stones(:,4),Fs] = audioread('Stones4.wav');
694 [Stones(:,5),Fs] = audioread('Stones5.wav');
695 [Stones(:,6),Fs] = audioread('Stones6.wav');
696 [Stones(:,7),Fs] = audioread('Stones7.wav');
697 [x8,Fs] = audioread('Stones8.wav');
698 [Stones(:,9),Fs] = audioread('Stones9.wav');

```

```

699 [Stones(:,10),Fs] = audioread('Stones10.wav');
700 [x11,Fs] = audioread('Stones11.wav');
701 [Stones(:,12),Fs] = audioread('Stones12.wav');
702 [x13,Fs] = audioread('Stones13.wav');
703 [Stones(:,14),Fs] = audioread('Stones14.wav');
704 [Stones(:,15),Fs] = audioread('Stones15.wav');
705 [Stones(:,16),Fs] = audioread('Stones16.wav');
706 [x17,Fs] = audioread('Stones17.wav');
707 [x18,Fs] = audioread('Stones18.wav');
708 [x19,Fs] = audioread('Stones19.wav');
709 [x20,Fs] = audioread('Stones20.wav');
710 [Stones(:,21),Fs] = audioread('Stones21.wav');
711 [x22,Fs] = audioread('Stones22.wav');
712 [Stones(:,23),Fs] = audioread('Stones23.wav');
713 [Stones(:,24),Fs] = audioread('Stones24.wav');
714 [Stones(:,25),Fs] = audioread('Stones25.wav');
715 [x26,Fs] = audioread('Stones26.wav');
716 [x27,Fs] = audioread('Stones27.wav');
717 [x28,Fs] = audioread('Stones28.wav');
718 [x29,Fs] = audioread('Stones29.wav');
719 [x30,Fs] = audioread('Stones30.wav');
720 Stones = Stones(1:220672,:);
721 Stones(:,1) = x1(1:220672);
722 Stones(:,8) = x8(1:220672);
723 Stones(:,11) = x11(1:220672);
724 Stones(:,13) = x13(1:220672);
725 Stones(:,17) = x17(1:220672);
726 Stones(:,18) = x18(1:220672);
727 Stones(:,19) = x19(1:220672);
728 Stones(:,20) = x20(1:220672);
729 Stones(:,22) = x22(1:220672);
730 Stones(:,26) = x26(1:220672);
731 Stones(:,27) = x27(1:220672);
732 Stones(:,28) = x28(1:220672);
733 Stones(:,29) = x29(1:220672);
734 Stones(:,30) = x30(1:220672);
735
736 [ACDC(:,1),Fs] = audioread('ACDC1.wav');
737 [ACDC(:,2),Fs] = audioread('ACDC2.wav');
738 [ACDC(:,3),Fs] = audioread('ACDC3.wav');
739 [ACDC(:,4),Fs] = audioread('ACDC4.wav');
740 [ACDC(:,5),Fs] = audioread('ACDC5.wav');
741 [ACDC(:,6),Fs] = audioread('ACDC6.wav');
742 [ACDC(:,7),Fs] = audioread('ACDC7.wav');
743 [z8,Fs] = audioread('ACDC8.wav');
744 [z9,Fs] = audioread('ACDC9.wav');

```

```

745 [z10,Fs] = audioread('ACDC10.wav');
746 [ACDC(:,11),Fs] = audioread('ACDC11.wav');
747 [z12,Fs] = audioread('ACDC12.wav');
748 [z13,Fs] = audioread('ACDC13.wav');
749 [ACDC(:,14),Fs] = audioread('ACDC14.wav');
750 [ACDC(:,15),Fs] = audioread('ACDC15.wav');
751 [ACDC(:,16),Fs] = audioread('ACDC16.wav');
752 [z17,Fs] = audioread('ACDC17.wav');
753 [ACDC(:,18),Fs] = audioread('ACDC18.wav');
754 [ACDC(:,19),Fs] = audioread('ACDC19.wav');
755 [ACDC(:,20),Fs] = audioread('ACDC20.wav');
756 [z21,Fs] = audioread('ACDC21.wav');
757 [ACDC(:,22),Fs] = audioread('ACDC22.wav');
758 [ACDC(:,23),Fs] = audioread('ACDC23.wav');
759 [z24,Fs] = audioread('ACDC24.wav');
760 [ACDC(:,25),Fs] = audioread('ACDC25.wav');
761 [ACDC(:,26),Fs] = audioread('ACDC26.wav');
762 [ACDC(:,27),Fs] = audioread('ACDC27.wav');
763 [z28,Fs] = audioread('ACDC28.wav');
764 [ACDC(:,29),Fs] = audioread('ACDC29.wav');
765 [ACDC(:,30),Fs] = audioread('ACDC30.wav');
766 ACDC = ACDC(1:220571,:);
767 ACDC(:,8) = z8(1:220571);
768 ACDC(:,9) = z9(1:220571);
769 ACDC(:,10) = z10(1:220571);
770 ACDC(:,12) = z12(1:220571);
771 ACDC(:,13) = z13(1:220571);
772 ACDC(:,17) = z17(1:220571);
773 ACDC(:,21) = z21(1:220571);
774 ACDC(:,24) = z24(1:220571);
775 ACDC(:,28) = z28(1:220571);
776
777 [AS(:,1),Fs] = audioread('Aerosmith1.wav');
778 [AS(:,2),Fs] = audioread('Aerosmith2.wav');
779 [AS(:,3),Fs] = audioread('Aerosmith3.wav');
780 [AS(:,4),Fs] = audioread('Aerosmith4.wav');
781 [AS(:,5),Fs] = audioread('Aerosmith5.wav');
782 [AS(:,6),Fs] = audioread('Aerosmith6.wav');
783 [AS(:,7),Fs] = audioread('Aerosmith7.wav');
784 [AS(:,8),Fs] = audioread('Aerosmith8.wav');
785 [AS(:,9),Fs] = audioread('Aerosmith9.wav');
786 [AS(:,10),Fs] = audioread('Aerosmith10.wav');
787 [AS(:,11),Fs] = audioread('Aerosmith11.wav');
788 [AS(:,12),Fs] = audioread('Aerosmith12.wav');
789 [AS(:,13),Fs] = audioread('Aerosmith13.wav');
790 [AS(:,14),Fs] = audioread('Aerosmith14.wav');

```

```

791 [AS(:,15),Fs] = audioread('Aerosmith15.wav');
792 [AS(:,16),Fs] = audioread('Aerosmith16.wav');
793 [AS(:,17),Fs] = audioread('Aerosmith17.wav');
794 [w18,Fs] = audioread('Aerosmith18.wav');
795 [AS(:,19),Fs] = audioread('Aerosmith19.wav');
796 [AS(:,20),Fs] = audioread('Aerosmith20.wav');
797 [AS(:,21),Fs] = audioread('Aerosmith21.wav');
798 [AS(:,22),Fs] = audioread('Aerosmith22.wav');
799 [AS(:,23),Fs] = audioread('Aerosmith23.wav');
800 [AS(:,24),Fs] = audioread('Aerosmith24.wav');
801 [AS(:,25),Fs] = audioread('Aerosmith25.wav');
802 [AS(:,26),Fs] = audioread('Aerosmith26.wav');
803 [AS(:,27),Fs] = audioread('Aerosmith27.wav');
804 [AS(:,28),Fs] = audioread('Aerosmith28.wav');
805 [AS(:,29),Fs] = audioread('Aerosmith29.wav');
806 [AS(:,30),Fs] = audioread('Aerosmith30.wav');
807 AS = AS(1:220160,:);
808 AS(:,18) = w18;
809
810
811 ACDC = ACDC(1:219547,:);
812 AS = AS(1:219547,:);
813 Stones = Stones(1:219547,:);
814 rock = [ACDC AS Stones];
815
816
817 Bentley = Bentley(1:219547,:);
818 Chesney = Chesney(1:219547,:);
819 Combs = Combs(1:219547,:);
820 country = [Bentley Chesney Combs];
821
822 Bach = Bach(1:219547,:);
823 Mozart = Mozart(1:219547,:);
824 YoYoMa = YoYoMa(1:219547,:);
825 classical = [Bach Mozart YoYoMa];
826 feature = 20;
827
828 Specclassical = zeros(262152,90);
829 Speccountry = zeros(262152,90);
830 Specrock = zeros(262152,90);
831 for jj = 1:90
832     S1 = spectrogram(classical(:,jj));
833     S2 = spectrogram(country(:,jj));
834     S3 = spectrogram(rock(:,jj));
835     Specclassical(:,jj) = abs(S1(:));
836     Speccountry(:,jj) = abs(S2(:));

```

```

837         Specrock(:,jj) = abs(S3(:));
838     end
839
840
841     [U,S,V,threshold1,threshold2,w,sortgenre1,sortgenre2,sortgenre3] =...
842         test3_trainer(Specclassical,Speccountry,Specrock,feature);
843
844     [Test(:,1),Fs] = audioread('ACDCTest1.wav');
845     [Test(:,2),Fs] = audioread('ACDCTest2.wav');
846     [Test(:,3),Fs] = audioread('ACDCTest3.wav');
847     [Test(:,4),Fs] = audioread('ACDCTest4.wav');
848     [Test(:,5),Fs] = audioread('ACDCTest5.wav');
849     [Test(:,6),Fs] = audioread('ACDCTest6.wav');
850     [Test(:,7),Fs] = audioread('ACDCTest7.wav');
851     [Test(:,8),Fs] = audioread('ASTest1.wav');
852     [Test(:,9),Fs] = audioread('ASTest2.wav');
853     [Test(:,10),Fs] = audioread('ASTest3.wav');
854     [Test(:,11),Fs] = audioread('ASTest4.wav');
855     [Test(:,12),Fs] = audioread('ASTest5.wav');
856     [Test(:,13),Fs] = audioread('ASTest6.wav');
857     [Test(:,14),Fs] = audioread('ASTest7.wav');
858     [Test15,Fs] = audioread('StonesTest1.wav');
859     [Test16,Fs] = audioread('StonesTest2.wav');
860     [Test17,Fs] = audioread('StonesTest3.wav');
861     [Test(:,18),Fs] = audioread('StonesTest4.wav');
862     [Test19,Fs] = audioread('StonesTest5.wav');
863     [Test(:,20),Fs] = audioread('StonesTest6.wav');
864     [Test(:,21),Fs] = audioread('StonesTest7.wav');
865     TestR = Test(1:220160,:);
866     TestR(:,15) = Test15(1:220160);
867     TestR(:,16) = Test16(1:220160);
868     TestR(:,17) = Test17(1:220160);
869     TestR(:,19) = Test19(1:220160);
870
871     [Test(:,1),Fs] = audioread('BachTest7.wav');
872     [Test2,Fs] = audioread('BachTest2.wav');
873     [Test3,Fs] = audioread('BachTest3.wav');
874     [Test4,Fs] = audioread('BachTest4.wav');
875     [Test5,Fs] = audioread('BachTest5.wav');
876     [Test6,Fs] = audioread('BachTest6.wav');
877     [Test7,Fs] = audioread('BachTest1.wav');
878     [Test(:,8),Fs] = audioread('MozartTest1.wav');
879     [Test(:,9),Fs] = audioread('MozartTest2.wav');
880     [Test(:,10),Fs] = audioread('MozartTest3.wav');
881     [Test(:,11),Fs] = audioread('MozartTest4.wav');
882     [Test(:,12),Fs] = audioread('MozartTest5.wav');

```



```

883 [Test(:,13),Fs] = audioread('MozartTest6.wav');
884 [Test(:,14),Fs] = audioread('MozartTest7.wav');
885 [Test(:,15),Fs] = audioread('YoYoMaTest1.wav');
886 [Test(:,16),Fs] = audioread('YoYoMaTest2.wav');
887 [Test(:,17),Fs] = audioread('YoYoMaTest3.wav');
888 [Test(:,18),Fs] = audioread('YoYoMaTest4.wav');
889 [Test(:,19),Fs] = audioread('YoYoMaTest5.wav');
890 [Test(:,20),Fs] = audioread('YoYoMaTest6.wav');
891 [Test(:,21),Fs] = audioread('YoYoMaTest7.wav');
892 TestClas = Test(1:220160,:);
893 TestClas(:,2) = Test2(1:220160);
894 TestClas(:,3) = Test3(1:220160);
895 TestClas(:,4) = Test4(1:220160);
896 TestClas(:,5) = Test5(1:220160);
897 TestClas(:,6) = Test6(1:220160);
898 TestClas(:,7) = Test7(1:220160);
899
900
901 [Test(:,1),Fs] = audioread('ChesneyTest1.wav');
902 [Test(:,2),Fs] = audioread('ChesneyTest2.wav');
903 [Test(:,3),Fs] = audioread('ChesneyTest3.wav');
904 [Test(:,4),Fs] = audioread('ChesneyTest4.wav');
905 [Test(:,5),Fs] = audioread('ChesneyTest5.wav');
906 [Test(:,6),Fs] = audioread('ChesneyTest6.wav');
907 [Test(:,7),Fs] = audioread('ChesneyTest7.wav');
908 [Test8,Fs] = audioread('CombsTest1.wav');
909 [Test9,Fs] = audioread('CombsTest2.wav');
910 [Test(:,10),Fs] = audioread('CombsTest3.wav');
911 [Test11,Fs] = audioread('CombsTest4.wav');
912 [Test12,Fs] = audioread('CombsTest5.wav');
913 [Test13,Fs] = audioread('CombsTest6.wav');
914 [Test14,Fs] = audioread('CombsTest7.wav');
915 [Test(:,15),Fs] = audioread('BentleyTest1.wav');
916 [Test(:,16),Fs] = audioread('BentleyTest2.wav');
917 [Test(:,17),Fs] = audioread('BentleyTest3.wav');
918 [Test(:,18),Fs] = audioread('BentleyTest4.wav');
919 [Test(:,19),Fs] = audioread('BentleyTest5.wav');
920 [Test(:,20),Fs] = audioread('BentleyTest6.wav');
921 [Test(:,21),Fs] = audioread('BentleyTest7.wav');
922 TestCount = Test(1:220160,:);
923 TestCount(:,8) = Test8(1:220160);
924 TestCount(:,9) = Test9(1:220160);
925 TestCount(:,11) = Test11(1:220160);
926 TestCount(:,12) = Test12(1:220160);
927 TestCount(:,13) = Test13(1:220160);
928 TestCount(:,14) = Test14(1:220160);

```

```

929
930 TestGenre = [TestClas TestCount TestR];
931
932 SpecTest = zeros(262152,63);
933 for jj = 1:63
934     S1 = spectrogram(TestGenre(:,jj));
935     SpecTest(:,jj) = abs(S1(:));
936 end
937 %%
938 TestMatrix = U'*SpecTest;
939 projection = w'*TestMatrix;
940 for j = 1:21
941     True(:,j) = [1;0];
942     True(:,j+21) = [0;0];
943     True(:,j+42) = [1;1];
944 end
945 ResVec(1,:) = (projection>threshold1);
946 ResVec(2,:) = (projection>threshold2)
947 True-ResVec
948 %% Plot classical/country/rock projections onto w
949 figure(3)
950 p = plot(sortgenre1,zeros(90),'ob','Linewidth',2)
951 for j = 2:90
952     set(get(get(p(j),'Annotation'),'LegendInformation'),...
953         'IconDisplayStyle','off');
954 end
955 hold on
956 p1 = plot(sortgenre2,.5*ones(90),'dr','Linewidth',2)
957 for j = 2:90
958     set(get(get(p1(j),'Annotation'),'LegendInformation'),...
959         'IconDisplayStyle','off');
960 end
961 p2 = plot(sortgenre3,ones(90),'*g','Linewidth',2)
962 for j = 2:90
963     set(get(get(p2(j),'Annotation'),'LegendInformation'),...
964         'IconDisplayStyle','off');
965 end
966 plot([threshold1 threshold1],[0 10],'r')
967 plot([threshold2 threshold2],[0 10],'r')
968 ylim([0 1.5])
969 title('Training Data for Classifier 3 (3 genres, 3 artists in each)',...
970     'FontSize',14)
971 legend('Genre 1 - Classical','Genre 2 - Country',...
972     'Genre 3 - Rock','threshold 1','threshold 2')
973 print(gcf,'Test3_dataplot.png','-dpng')
974 %% Functions

```

```

975 function [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] ...
976 = test1_trainer(artist1,artist2,artist3,feature)
977 size_art1 = size(artist1,2);
978 size_art2 = size(artist2,2);
979 size_art3 = size(artist3,2);
980 [U,S,V] = svd([artist1 artist2 artist3], 'econ');
981 artists = S*V'; % projection onto principal components
982 U = U(:,1:feature);
983 artist1songs = artists(1:feature,1:size_art1);
984 artist2songs = artists(1:feature,size_art1+1:size_art1+size_art2);
985 artist3songs = artists(1:feature,size_art1+size_art2+1:size_art1+...
986     size_art2+size_art3);
987 mean_art1 = mean(artist1songs,2);
988 mean_art2 = mean(artist2songs,2);
989 mean_art3 = mean(artist3songs,2);
990 Sw = 0; % within class variances
991 for k=1:size_art1
992     Sw = Sw + (artist1songs(:,k)-mean_art1)*(artist1songs(:,k)-mean_art1)';
993 end
994 for k=1:size_art2
995     Sw = Sw + (artist2songs(:,k)-mean_art2)*(artist2songs(:,k)-mean_art2)';
996 end
997 for k=1:size_art3
998     Sw = Sw + (artist3songs(:,k)-mean_art3)*(artist3songs(:,k)-mean_art3)';
999 end
1000
1001 sample_mean = (mean_art1+mean_art2+mean_art3)/3;
1002 Sb = 30*(mean_art1-sample_mean)*(mean_art1-sample_mean)';
1003 Sb = Sb + 30*(mean_art2-sample_mean)*(mean_art2-sample_mean)';
1004 Sb = Sb + 30*(mean_art3-sample_mean)*(mean_art3-sample_mean)'; % between class
1005 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
1006 [~,ind] = max(abs(diag(D)));
1007 w = V2(:,ind); w = w/norm(w,2);
1008 v_art1 = w'*artist1songs;
1009 v_art2 = w'*artist2songs;
1010 v_art3 = w'*artist3songs;
1011 %art1 on left and art3 on right
1012 if mean(v_art3)>mean(v_art2)
1013     w = -w;
1014     v_art2 = -v_art2;
1015     v_art3 = -v_art3;
1016 end
1017 %art1 < threshold1 < art2 < threshold2 < art3
1018 sortart1 = sort(v_art1);
1019 sortart2 = sort(v_art2);
1020 sortart3 = sort(v_art3);

```

```

1021     t1 = length(sortart1);
1022     t2 = 1;
1023     while sortart1(t1)>sortart3(t2)
1024         t1 = t1-1;
1025         t2 = t2+1;
1026     end
1027     threshold1 = (sortart1(t1)+sortart3(t2))/2;
1028
1029     t1 = length(sortart2);
1030     t2 = 1;
1031     while sortart3(t1)>sortart2(t2)
1032         t1 = t1-1;
1033         t2 = t2+1;
1034     end
1035     threshold2 = (sortart3(t1)+sortart2(t2))/2;
1036 end
1037
1038 function [U,S,V,threshold1,threshold2,w,sortart1,sortart2,sortart3] =...
1039     test2_trainer(artist1,artist2,artist3,feature)
1040 size_art1 = size(artist1,2);
1041 size_art2 = size(artist2,2);
1042 size_art3 = size(artist3,2);
1043 [U,S,V] = svd([artist1 artist2 artist3],'econ');
1044 artists = S*V'; % projection onto principal components
1045 U = U(:,1:feature);
1046 artist1songs = abs(artists(1:feature,1:size_art1));
1047 artist2songs = abs(artists(1:feature,size_art1+1:size_art1+size_art2));
1048 artist3songs = abs(artists...
1049     (1:feature,size_art1+size_art2+1:size_art1+size_art2+size_art3));
1050 mean_art1 = mean(artist1songs,2);
1051 mean_art2 = mean(artist2songs,2);
1052 mean_art3 = mean(artist3songs,2);
1053 Sw = 0; % within class variances
1054 for k=1:size_art1
1055     Sw = Sw + (artist1songs(:,k)-mean_art1)*(artist1songs(:,k)-mean_art1)';
1056 end
1057 for k=1:size_art2
1058     Sw = Sw + (artist2songs(:,k)-mean_art2)*(artist2songs(:,k)-mean_art2)';
1059 end
1060 for k=1:size_art3
1061     Sw = Sw + (artist3songs(:,k)-mean_art3)*(artist3songs(:,k)-mean_art3)';
1062 end
1063
1064 sample_mean = (mean_art1+mean_art2+mean_art3)/3;
1065 Sb = 30*(mean_art1-sample_mean)*(mean_art1-sample_mean)';
1066 Sb = Sb + 30*(mean_art2-sample_mean)*(mean_art2-sample_mean)';

```

```

1067 Sb = Sb + 30*(mean_art3-sample_mean)*(mean_art3-sample_mean)'; % between class
1068 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
1069 [~,ind] = max(abs(diag(D)));
1070 w = V2(:,ind); w = w/norm(w,2);
1071 v_art1 = w'*artist1songs;
1072 v_art2 = w'*artist2songs;
1073 v_art3 = w'*artist3songs;
1074 %art1 on left and art3 on right
1075 if mean(v_art2)>mean(v_art1)
1076     w = -w;
1077     v_art2 = -v_art2;
1078     v_art1 = -v_art1;
1079 end
1080 %art1 < threshold1 < art2 < threshold2 < art3
1081 sortart1 = sort(v_art1);
1082 sortart2 = sort(v_art2);
1083 sortart3 = sort(v_art3);
1084 t1 = length(sortart1);
1085 t2 = 1;
1086 while sortart1(t1)>sortart2(t2)
1087     t1 = t1-1;
1088     t2 = t2+1;
1089 end
1090 threshold1 = (sortart1(t1)+sortart2(t2))/2;
1091
1092 t1 = length(sortart2);
1093 t2 = 1;
1094 while sortart1(t1)>sortart3(t2)
1095     t1 = t1-1;
1096     t2 = t2+1;
1097 end
1098 threshold2 = (sortart1(t1)+sortart3(t2))/2;
1099 end
1100
1101
1102 function [U,S,V,threshold1,threshold2,w,sortgenre1,sortgenre2,sortgenre3] =...
1103     test3_trainer(genre1,genre2,genre3,feature)
1104 size_genre1 = size(genre1,2);
1105 size_genre2 = size(genre2,2);
1106 size_genre3 = size(genre3,2);
1107 [U,S,V] = svd([genre1 genre2 genre3],'econ');
1108 genres = S*V'; % projection onto principal components
1109 U = U(:,1:feature);
1110 genre1songs = genres(1:feature,1:size_genre1);
1111 genre2songs = genres(1:feature,size_genre1+1:size_genre1+size_genre2);
1112 genre3songs = genres(1:feature,size_genre1+size_genre2+1:size_genre1+...

```

```

1113     size_genre2+size_genre3);
1114 mean_genre1 = mean(genre1songs,2);
1115 mean_genre2 = mean(genre2songs,2);
1116 mean_genre3 = mean(genre3songs,2);
1117 Sw = 0; % within class variances
1118 for k=1:size_genre1
1119     Sw = Sw + (genre1songs(:,k)-mean_genre1)*(genre1songs(:,k)-mean_genre1)';
1120 end
1121 for k=1:size_genre2
1122     Sw = Sw + (genre2songs(:,k)-mean_genre2)*(genre2songs(:,k)-mean_genre2)';
1123 end
1124 for k=1:size_genre3
1125     Sw = Sw + (genre3songs(:,k)-mean_genre3)*(genre3songs(:,k)-mean_genre3)';
1126 end
1127
1128 sample_mean = (mean_genre1+mean_genre2+mean_genre3)/3;
1129 Sb = 90*(mean_genre1-sample_mean)*(mean_genre1-sample_mean)';
1130 Sb = Sb + 90*(mean_genre2-sample_mean)*(mean_genre2-sample_mean)';
1131 Sb = Sb + 90*(mean_genre3-sample_mean)*...
1132     (mean_genre3-sample_mean)'; % between class
1133 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
1134 [~,ind] = max(abs(diag(D)));
1135 w = V2(:,ind); w = w/norm(w,2);
1136 v_genre1 = w'*genre1songs;
1137 v_genre2 = w'*genre2songs;
1138 v_genre3 = w'*genre3songs;
1139 %art1 on left and art3 on right
1140 if mean(v_genre1)>mean(v_genre3)
1141     w = -w;
1142     v_genre3 = -v_genre3;
1143     v_genre1 = -v_genre1;
1144 end
1145 %art1 < threshold1 < art2 < threshold2 < art3
1146 sortgenre1 = sort(v_genre1);
1147 sortgenre2 = sort(v_genre2);
1148 t1 = length(sortgenre1);
1149 t2 = 1;
1150 while sortgenre2(t1)>sortgenre1(t2)
1151     t1 = t1-1;
1152     t2 = t2+1;
1153 end
1154 threshold1 = (sortgenre2(t1)+sortgenre1(t2))/2;
1155 %threshold1 = -710.165;
1156
1157 sortgenre3 = sort(v_genre3)
1158 t2 = length(sortgenre2);

```

```

1159     t3 = 1;
1160     while sortgenre1(t2)>sortgenre3(t3)
1161         t2 = t2-1
1162         t3 = t3+1
1163     end
1164     threshold2 = abs((sortgenre1(t2)+sortgenre3(t3))/2);
1165     %threshold2 = 1692.165;
1166 end

```