

# Music Classification Using Linear Discriminant Analysis

Christina Smith

AMATH 482 - Winter 2020

## Abstract

In this exercise, we create algorithms to classify audio clips. We break a data set into training data and test data and use principal component analysis and linear discriminant analysis to create and evaluate a model to classify new audio clips.

## I. Introduction

Our goal is to classify data in three different tests:

- Test 1: Classify three different bands of different genres.  
(*Rubblebucket, Nina Simone, Air*)
- Test 2: Classify three different bands from the same genre.  
(*Al Green, Otis Redding, Sam Cooke*)
- Test 1: Classify three different genres.  
(*Chill Beats, Country, Rock*)

Using singular value decomposition(SVD) and principal component analysis(PCA) we evaluate a small subset of our data (the training set). We chose a value for the number of features(principal components) we would like to evaluate. Next we determine the variance within classes and the covariance for the means of the different classes.

By performing eigenvalue decomposition on these matrices we can determine the bases which give the minimum variance within classes and the maximum covariance for the means of the different classes when we project the data onto it. Since we have three different classes, we can choose a 1 or 2 dimensional basis. If we are evaluating the data on a line, we calculate the thresholds on the line that give us the best separation of the three classes. If we are evaluating in a plane, we compute the mean coordinates of each cluster of data and determine class by finding the minimum distance of a given test sample to the means of the three classes.

## II. Theoretical Background

### SVD and PCA

Since we want to determine a way to classify data from our three different groups for each test, we will combine our data and use SVD and PCA as way to determine the basis with the most variance, that is, the principal component vectors with the highest corresponding singular values. In this case, the rank of this basis is also described as the number of 'features' we want to evaluate the data for.

When we have chosen the principal component vectors, we project our data onto the new basis.

### Linear Discriminant Analysis

Now that we have a basis with the maximum variance of our complete data set we want to determine another basis to project this data onto such that there is minimal overlap in the data from each class. The way to do this is to determine a basis of projection that maximizes the covariance between each class mean and minimizes the variance within each class (in regards to each dimension of the projection basis).

We first create a matrix of the variance within individual sets of data using the following formula:

$$\mathbf{S}_W = \sum_{j=1}^N \sum_{\mathbf{x}} (\mathbf{x} - \boldsymbol{\mu}_j)(\mathbf{x} - \boldsymbol{\mu}_j)^T \quad (1)$$

Where  $N$  is the number of classes,  $\boldsymbol{\mu}$  is a vector which contains the mean of each row in the submatrix containing the data of a given class and  $\mathbf{x}$  indicates we perform this process for every column in the submatrix.

The matrix of the covariance between the means of the sets of data is computed as follows:

$$\mathbf{S}_B = \sum_{j=1}^N (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T \quad (2)$$

Where  $\boldsymbol{\mu}$  contains the mean across all classes. Using these matrices we compute the eigenvalue decomposition

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \quad (3)$$

Because our data set contains three classes,  $\mathbf{w}$  is a matrix which contains the vectors that represent the basis that maximizes the covariance between each class mean and minimizes the variance within each class.

## Classification

With the training data projected onto the new basis a model is formed to differentiate the classes. One method is to evaluate the means of the clusters of data on a line and to set threshold values which will predict whether test data belongs to a given class. Another method is to project the training data onto a planar basis of the two vectors of  $\mathbf{w}$  and then evaluate the mean location of each cluster. Test data will be projected on the same plane and evaluated based on its distance from the means of the class data.

## III. Algorithm Implementation and Development

### Data Acquisition (See App. A: 'get\_data.m' and 'create\_spectrograms.m')

Audio data was processed into wav files each containing 5 seconds of music. For test 1, samples were taken of 4 songs with 10 clips of each song. For tests 2 and 3 samples were taken of 7 songs with 10 clips of each song. This data was then used create spectrograms. The spectrograms had a range in frequency of +/- 4000 Hz in order to capture the most relevant data.

### trainer() (See App. A: 'HW\_04.m' lines 300-490)

This function receives the spectrogram data for the three classes, as well as information about the number of features to evaluate. All of the spectrogram data is combined into a single matrix and then SVD is performed on this matrix. The data is then projected on the principal component basis and the matrix is separated into three separate matrices with the number of rows corresponding to the number of features.

The variance and covariance matrices are computed using Formulas 1 and 2 respectively and SVD is performed using Matlab's eig() function. The eigenvectors corresponding to the eigenvalues with the maximum absolute value are normalized. These unit vectors are the basis for our analysis.

The training data is projected onto these vectors and the means of each class are determined. These means are used to determine where each class falls on the projection basis (maximum, middle value, minimum). One simple way of determining the two thresholds is to find the mean of each class on the projection line the halfway points between the max and middle mean, and between the middle and min mean are used as the threshold values. Another possible method, given that we have three classes, is to project the training data onto the  $\mathbf{w1}$  and  $\mathbf{w2}$  coordinate plane and determining the coordinates for the minimum of each cluster. When test data is evaluated, these mean coordinates will be used to determine the class of a given test sample.

The function returns the matrices from the SVD of the training data, as well as the  $\mathbf{w1}$ ,  $\mathbf{w2}$  vectors, the corresponding eigenvalues, the thresholds, the rank of each class and the projection of the data onto the  $\mathbf{w1}$ ,  $\mathbf{w2}$  vectors.

**classify1D()** (See App.A: 'HW\_04.m' lines 254-269)

This function takes the projected values (on the **w1** line) of the test data, the threshold values for **w1**, and the rank of each class. It then determines where each value falls in relation to the thresholds and then assigns a class based on the rank information. It returns a vector of classes corresponding to the projected value of each sample.

**classify2D()** (See App. A: 'HW\_04.m' lines 276-297)

This function takes the projected values (on the **w1, w2** plane) of the test data, the mean coordinate values of the training data. It then determines which mean is the closest for every sample, and assigns a class based on the mean with the minimum distance. It returns a vector of classes corresponding to the projected value of each sample.

## IV. Computational Results & Supplementary Plots

### Test 1 - Three different bands of different genres (See Fig. 1)

With features set to 50, a training set of 25 samples and a test set of 15 samples, the prediction accuracies using both 1D and 2D classification were as follows:

	Class A	Class B	Class C
1D	86%	20%	53%
2D	100%	20%	73%

Class B was the lowest range of values on the **w1** projection, Class A was the middle range, and Class C was the upper range. The threshold values were -4.1 and 5.3.

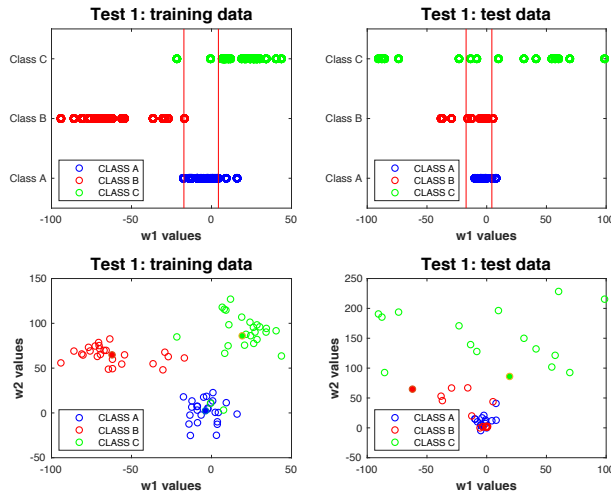


Figure 1: Test 1: Projections of Training and Test Data

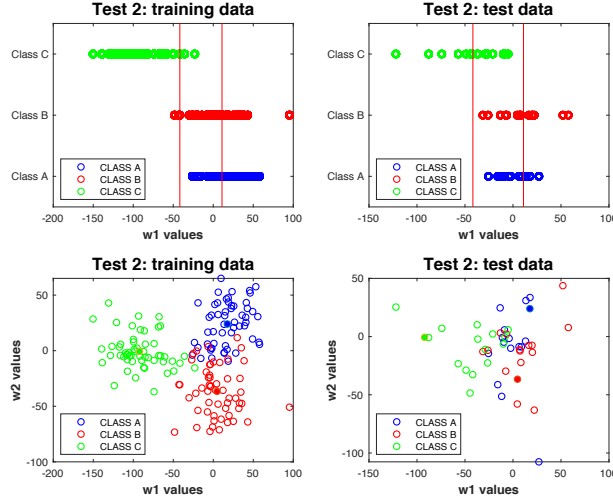


Figure 2: Test 2: Projections of Training and Test Data

### Test 2 - Three different bands from the same genre (See Fig. 2)

With features set to 80, a training set of 55 samples and a test set of 15 samples, the prediction accuracies using both 1D and 2D classification were as follows:

	Class A	Class B	Class C
1D	26.7%	60%	46.7%
2D	46.7%	66.7%	40%

Class C was the lowest range of values on the  $w_1$  projection, Class B was the middle range, and Class A was the upper range. The threshold values were -41.8 and 10.9.

### Test 3 - Three different genres. (See Fig. 3)

With features set to 80, a training set of 55 samples and a test set of 15 samples, the prediction accuracies using both 1D and 2D classification were as follows:

	Class A	Class B	Class C
1D	73.3%	20%	40%
2D	66.7%	33.3%	46.7%

Class C was the lowest range of values on the  $w_1$  projection, Class B was the middle range, and Class A was the upper range. The threshold values were -41.8 and 10.9.

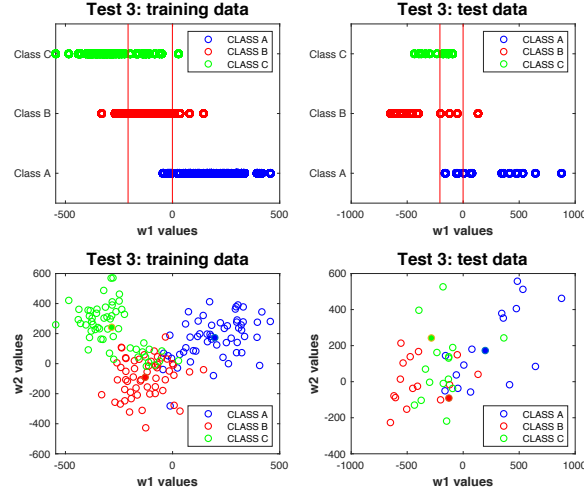


Figure 3: Test 3: Projections of Training and Test Data

## V. Summary and Conclusions

From reviewing the accuracy percentages it appears that evaluating the data using both projection vectors *generally* leads to more accurate classification. It is worth noting that in Tests 2 and 3, the accuracy of one class did go down with prediction in 2 dimensions, however the accuracy gained in predicting each of the other two classes was greater than the accuracy lost by the one class.

The classifications were most accurate on average in the first test. This follows what I was expecting, because the samples all differed in both artist and genre. The second and third tests had lower average accuracy. This made sense, since these data sets had more sound in common than in Test 1.

Changing the parameters of the feature number and the size of the training and sample sets also had an effect. For example, in Test 3, changing the number of features from 80 to 5 resulted in a 0% accuracy rate for Class B (in 2D). A smaller number of features resulted in a larger variance in the data and a larger number of features had the opposite effect. Changing the size of the training data set from 55 to 10, decreased the accuracy in predicting Class A by almost half. These also affected the spread of the data on the projected line or plane.

The classification model itself can also be optimized. While the methods used here are fairly simple, one possibility would be to evaluate the data in 2D giving more weight to the relationship between means along the  $\mathbf{w1}$  axis, as that is the basis that has best minimized the in-class variance and maximized the between class variance. There is much more to explore in this type of modelling, as well as how the parameters and the data sets themselves affect the accuracy of classification.

## Appendix A: Matlab Code

### get\_data.m

```

1  %%% Test 3 c
2  % close all; clear all; clc;
3  %
4  % s = (50000)*2;
5  %
6  % dat_3c = zeros(s,70); % 40 samples total (10/song)
7  %
8  % count = 0;
9  % for i=1:7
10 %     % go through each song
11 %     filename = strcat('test_03/rock/all_songs/0',num2str(i),'.
    mp3');
12 %     [y,Fs] = audioread(filename);
13 %     y_mono = y(:,1); % mono
14 %     b = 1:2:length(y_mono); % resampling vector
15 %     y_resampled = y_mono(b)'; % transpose the y vector
16 %
17 %     % get 10 samples of each song
18 %     for j=1:10
19 %         dat_3c(:,count+1) = y_resampled(1:s); % new 5 second
    clip
20 %         last = size(y_resampled);
21 %         y_resampled = y_resampled(s+1:last(2));
22 %         if count > 9
23 %             wavname =...
24 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
    nosync/Homework/HW_04/test_03/rock/all_clips/0',...
25 %                 num2str(count),'.wav');
26 %             audiowrite(wavname,dat_3c(:,count+1),Fs/2);
27 %         else
28 %             wavname =...
29 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
    nosync/Homework/HW_04/test_03/rock/all_clips/00',...
30 %                 num2str(count),'.wav');
31 %             audiowrite(wavname,dat_3c(:,count+1),Fs/2);
32 %         end
33 %         count = count+1;
34 %     end
35 % end
36 %
37 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_03/test_03c.mat dat_3c
38
39 %%% Test 3 b
40 % close all; clear all; clc;
41 %
42 % s = (50000)*2;
43 %
44 % dat_3b = zeros(s,70); % 70 samples total (10/song)
45 %

```

```

46 % count = 0;
47 % for i=1:7
48 %     % go through each song
49 %     filename = strcat('test_03/country/all_songs/0',num2str(i)
    ,'.mp3');
50 %     [y,Fs] = audioread(filename);
51 %     y_mono = y(:,1); % mono
52 %     b = 1:2:length(y_mono); % resampling vector
53 %     y_resampled = y_mono(b)'; % transpose the y vector
54 %
55 %     % get 10 samples of each song
56 %     for j=1:10
57 %         dat_3b(:,count+1) = y_resampled(1:s); % new 5 second
clip
58 %         last = size(y_resampled);
59 %         y_resampled = y_resampled(s+1:last(2));
60 %         if count > 9
61 %             wavname =...
62 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_03/country/all_clips/0',...
63 %                 num2str(count),'.wav');
64 %             audiowrite(wavname,dat_3b(:,count+1),Fs/2);
65 %         else
66 %             wavname =...
67 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_03/country/all_clips/00',...
68 %                 num2str(count),'.wav');
69 %             audiowrite(wavname,dat_3b(:,count+1),Fs/2);
70 %         end
71 %         count = count+1;
72 %     end
73 % end
74 %
75 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_03/test_03b.mat dat_3b
76 % %
77 % %% Test 3 a
78 % close all; clear all; clc;
79 %
80 % s = (50000)*2;
81 %
82 % dat_3a = zeros(s,40); % 40 samples total (10/song)
83 %
84 % count = 0;
85 % for i=1:7
86 %     % go through each song
87 %     filename = strcat('test_03/chill_beats/all_songs/0',num2str(
    i),'.mp3');
88 %     [y,Fs] = audioread(filename);
89 %     y_mono = y(:,1); % mono
90 %     b = 1:2:length(y_mono); % resampling vector
91 %     y_resampled = y_mono(b)'; % transpose the y vector
92 %
93 %     % get 10 samples of each song

```



```

94 %         for j=1:10
95 %             dat_3a(:,count+1) = y_resampled(1:s);    % new 5 second
           clip
96 %             last = size(y_resampled);
97 %             y_resampled = y_resampled(s+1:last(2));
98 %             if count > 9
99 %                 wavname = ...
100 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_03/chill_beats/all_clips/0',...
101 %                 num2str(count),'.wav');
102 %                 audiowrite(wavname,dat_3a(:,count+1),Fs/2);
103 %             else
104 %                 wavname = ...
105 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_03/chill_beats/all_clips/00',...
106 %                 num2str(count),'.wav');
107 %                 audiowrite(wavname,dat_3a(:,count+1),Fs/2);
108 %             end
109 %             count = count+1;
110 %         end
111 % end
112 %
113 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
HW_04/test_03/test_03a.mat dat_3a
114 %
115 %
116 % %% Test 2 c
117
118 % close all; clear all; clc;
119 %
120 % s = (50000)*2;
121 %
122 % dat_2c = zeros(s,70); % 70 samples total (10/song)
123 %
124 % count = 0;
125 % for i=1:7
126 %     % go through each song
127 %     filename = strcat('test_02/sam-cooke/all_songs/0',num2str(i)
, '.wav');
128 %     [y,Fs] = audioread(filename);
129 %     y_mono = y(:,1); % mono
130 %     b = 1:2:length(y_mono); % resampling vector
131 %     y_resampled = y_mono(b)'; % transpose the y vector
132 %
133 %     % get 10 samples of each song
134 %     for j=1:10
135 %         dat_2c(:,count+1) = y_resampled(1:s);    % new 5 second
           clip
136 %         last = size(y_resampled);
137 %         y_resampled = y_resampled(s+1:last(2));
138 %         if count > 9
139 %             wavname = ...
140 %             strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_02/sam-cooke/all_clips/0',...

```

```

141 %             num2str(count), '.wav');
142 %             audiowrite(wavname, dat_2c(:, count+1), Fs/2);
143 %         else
144 %             wavname = ...
145 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_02/sam-cooke/all_clips/00', ...
146 %                 num2str(count), '.wav');
147 %             audiowrite(wavname, dat_2c(:, count+1), Fs/2);
148 %         end
149 %         count = count+1;
150 %     end
151 % end
152 %
153 %
154 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
HW_04/test_02/test_02c.mat dat_2c
155 %
156 %% Test 2 b
157
158 % close all; clear all; clc;
159 %
160 % s = (50000)*2;
161 %
162 % dat_2b = zeros(s,70); % 70 samples total (10/song)
163 %
164 % count = 0;
165 % for i=1:7
166 %     % go through each song
167 %     filename = strcat('test_02/otis-redding/all_songs/0', num2str
(i), '.wav');
168 %     [y, Fs] = audioread(filename);
169 %     y_mono = y(:,1); % mono
170 %     b = 1:2:length(y_mono); % resampling vector
171 %     y_resampled = y_mono(b)'; % transpose the y vector
172 %
173 %     % get 10 samples of each song
174 %     for j=1:10
175 %         dat_2b(:, count+1) = y_resampled(1:s); % new 5 second
clip
176 %         last = size(y_resampled);
177 %         y_resampled = y_resampled(s+1:last(2));
178 %         if count > 9
179 %             wavname = ...
180 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_02/otis-redding/all_clips/0', ...
181 %                 num2str(count), '.wav');
182 %             audiowrite(wavname, dat_2b(:, count+1), Fs/2);
183 %         else
184 %             wavname = ...
185 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_02/otis-redding/all_clips/00', ...
186 %                 num2str(count), '.wav');
187 %             audiowrite(wavname, dat_2b(:, count+1), Fs/2);
188 %         end

```

```

189 %             count = count+1;
190 %         end
191 % end
192 %
193 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
      HW_04/test_02/test_02b.mat dat_2b
194
195
196 %%% Test 2 a
197 %
198 % close all; clear all; clc;
199 %
200 % s = (50000)*2;
201 %
202 % dat_2a = zeros(s,40); % 40 samples total (10/song)
203 %
204 % count = 0;
205 % for i=1:7
206 %     % go through each song
207 %     filename = strcat('test_02/al_green/all_songs/0',num2str(i)
      ,'.wav');
208 %     [y,Fs] = audioread(filename);
209 %     y_mono = y(:,1); % mono
210 %     b = 1:2:length(y_mono); % resampling vector
211 %     y_resampled = y_mono(b)'; % transpose the y vector
212 %
213 %     % get 10 samples of each song
214 %     for j=1:10
215 %         dat_2a(:,count+1) = y_resampled(1:s); % new 5 second
      clip
216 %         last = size(y_resampled);
217 %         y_resampled = y_resampled(s+1:last(2));
218 %         if count > 9
219 %             wavname =...
220 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
      nosync/Homework/HW_04/test_02/al_green/all_clips/0',...
221 %                     num2str(count),'.wav');
222 %             audiowrite(wavname,dat_2a(:,count+1),Fs/2);
223 %         else
224 %             wavname =...
225 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
      nosync/Homework/HW_04/test_02/al_green/all_clips/00',...
226 %                     num2str(count),'.wav');
227 %             audiowrite(wavname,dat_2a(:,count+1),Fs/2);
228 %         end
229 %         count = count+1;
230 %     end
231 % end
232 %
233 %
234 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
      HW_04/test_02/test_02a.mat dat_2a
235 %
236 %

```

```

237 % %% Test 1 c
238 %
239 % close all; clear all; clc;
240 %
241 % s = (50000)*2;
242 %
243 % dat_1c = zeros(s,40); % 40 samples total (10/song)
244 %
245 % count = 0;
246 % for i=1:4
247 %     % go through each song
248 %     filename = strcat('test_01/air/all_songs/0',num2str(i),'.wav
    ');
249 %     [y,Fs] = audioread(filename);
250 %     y_mono = y(:,1); % mono
251 %     b = 1:2:length(y_mono); % resampling vector
252 %     y_resampled = y_mono(b)'; % transpose the y vector
253 %
254 %     % get 10 samples of each song
255 %     for j=1:10
256 %         dat_1c(:,count+1) = y_resampled(1:s); % new 5 second
clip
257 %         last = size(y_resampled);
258 %         y_resampled = y_resampled(s+1:last(2));
259 %         if count > 9
260 %             wavname =...
261 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/air/all_clips/0',...
262 %                 num2str(count),'.wav');
263 %             audiowrite(wavname,dat_1c(:,count+1),Fs/2);
264 %         else
265 %             wavname =...
266 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/air/all_clips/00',...
267 %                 num2str(count),'.wav');
268 %             audiowrite(wavname,dat_1c(:,count+1),Fs/2);
269 %         end
270 %         count = count+1;
271 %     end
272 % end
273 %
274 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_01/test_01c.mat dat_1c
275 %
276 % %% Test 1 b
277 % close all; clear all; clc;
278 %
279 % s = (50000)*2;
280 %
281 % dat_1b = zeros(s,40); % 40 samples total (10/song)
282 %
283 % count = 0;
284 % for i=1:4
285 %     % go through each song

```

```

286 %     filename = strcat('test_01/nina/all_songs/0',num2str(i),'.
      wav');
287 %     [y,Fs] = audioread(filename);
288 %     y_mono = y(:,1); % mono
289 %     b = 1:2:length(y_mono); % resampling vector
290 %     y_resampled = y_mono(b)'; % transpose the y vector
291 %
292 %     % get 10 samples of each song
293 %     for j=1:10
294 %         dat_1b(:,count+1) = y_resampled(1:s); % new 5 second
      clip
295 %         last = size(y_resampled);
296 %         y_resampled = y_resampled(s+1:last(2));
297 %         if count > 9
298 %             wavname =...
299 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
      nosync/Homework/HW_04/test_01/nina/all_clips/0',...
300 %                 num2str(count),'.wav');
301 %                 audiowrite(wavname,dat_1b(:,count+1),Fs/2);
302 %         else
303 %             wavname =...
304 %                 strcat('/Users/christinasmith/Desktop/AMATH482.
      nosync/Homework/HW_04/test_01/nina/all_clips/00',...
305 %                 num2str(count),'.wav');
306 %                 audiowrite(wavname,dat_1b(:,count+1),Fs/2);
307 %         end
308 %         count = count+1;
309 %
310 %     end
311 % end
312 %
313 %
314 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
      HW_04/test_01/test_01b.mat dat_1b
315 %
316 % %% Test 1 a
317 %
318 %
319 % close all; clear all; clc;
320 %
321 % s = (50000)*2;
322 %
323 %
324 % dat_1a = zeros(s,40); % 40 samples total (10/song)
325 %
326 % count = 0;
327 % for i=1:3
328 %     % Users/christinasmith/Desktop/AMATH482.nosync/Homework/
      HW_04/test_01/air/all_songs/01.wav
329 %     % go through each song
330 %     filename = strcat('test_01/rubblebucket/all_songs/0',num2str
      (i),'.wav');
331 %     [y,Fs] = audioread(filename);
332 %     y_mono = y(:,1); % mono

```

```

333 %      b = 1:2:length(y_mono); % resampling vector
334 %      y_resampled = y_mono(b)'; % transpose the y vector
335 %
336 %      % get 10 samples of each song
337 %      for j=1:10
338 %          dat_1a(:,count+1) = y_resampled(1:s); % new 5 second
clip
339 %          last = size(y_resampled);
340 %          y_resampled = y_resampled(s+1:last(2));
341 %          if count > 9
342 %              wavname = ...
343 %                  strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/rubblebucket/all_clips/0',...
344 %                      num2str(count), '.wav');
345 %                      audiowrite(wavname, dat_1a(:,count+1), Fs/2);
346 %          else
347 %              wavname = ...
348 %                  strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/rubblebucket/all_clips/00',...
349 %                      num2str(count), '.wav');
350 %                      audiowrite(wavname, dat_1a(:,count+1), Fs/2);
351 %          end
352 %          count = count+1;
353 %      end
354 % end
355 %
356 %
357 % filename = strcat('test_01/rubblebucket/all_songs/la.wav');
358 % [y,Fs] = audioread(filename);
359 % y_mono = y(:,1); % mono
360 % b = 1:2:length(y_mono); % resampling vector
361 % y_resampled = y_mono(b)'; % transpose the y vector
362 %
363 % for j=1:9
364 %     dat_1a(:,count+1) = y_resampled(1:s); % new 5 second clip
365 %     last = size(y_resampled);
366 %     y_resampled = y_resampled(s+1:last(2));
367 %     if count > 9
368 %         wavname = ...
369 %             strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/rubblebucket/all_clips/0',...
370 %                 num2str(count), '.wav');
371 %                 audiowrite(wavname, dat_1a(:,count+1), Fs/2);
372 %     else
373 %         wavname = ...
374 %             strcat('/Users/christinasmith/Desktop/AMATH482.
nosync/Homework/HW_04/test_01/rubblebucket/all_clips/00',...
375 %                 num2str(count), '.wav');
376 %                 audiowrite(wavname, dat_1a(:,count+1), Fs/2);
377 %     end
378 %     count = count+1;
379 % end
380 %
381 %

```

```

382 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_01/test_01a.mat dat_1a

create_spectrograms.m

1 %% Test 3 c
2
3 close all; clear all; clc;
4
5 load test_03/test_03c.mat
6
7
8 % set the time and frequency domains
9
10 size = size(dat_3c);
11 n=size(2); % number of samples
12 s = size(1); % length of samples
13 L=s/(44100/2); % time domain
14 t2=linspace(0,L,s+1);
15 t=t2(1:s);
16 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
17 ks_Hz=fftshift(k_Hz);
18 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
19
20 a = 2e4;
21 tstep=0:0.3:L;
22
23
24 % array of spectrogram data
25 sp_3c = zeros(8001*length(tstep),n);
26 spect_array = zeros(8001,length(tstep),n);
27
28 for j=1:n
29     % spectrogram of clip
30     spect = zeros(s,length(tstep));
31     for k=1:length(tstep)
32         g=exp(-a*(t-tstep(k)).^2);
33         Sg=g'.* dat_3c(:,j); % apply new filter to the current 5sec
            clip
34         Sgt=fft(Sg); % transform to frequency domain
35         spect(:,k) = fftshift(abs(Sgt));
36     end
37     spect = spect((s/2-4000):(s/2+4000),:);
38     spect_array(:,j) = spect;
39     sp_3c(:,j) = reshape(spect,[8001*length(tstep),1]);
40 end
41
42 figure(1)
43 subplot(1,1,1)
44 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
45 shading interp
46 title('Home-made a = 1e03, inc = 0.0446','FontSize',16)
47 set(gca,'FontSize',11)
48 colormap(hot)

```

```

49
50 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_03/test_03c-sp.mat sp_3c
51
52 %% Test 3 b
53
54 close all; clear all; clc;
55
56 load test_03/test_03b.mat
57
58
59 % set the time and frequency domains
60
61 size = size(dat_3b);
62 n=size(2); % number of samples
63 s = size(1); % length of samples
64 L=s/(44100/2); % time domain
65 t2=linspace(0,L,s+1);
66 t=t2(1:s);
67 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
68 ks_Hz=fftshift(k_Hz);
69 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
70
71 a = 2e4;
72 tstep=0:0.3:L;
73
74
75 % array of spectrogram data
76 sp_3b = zeros(8001*length(tstep),n);
77 spect_array = zeros(8001,length(tstep),n);
78
79 for j=1:n
80     % spectrogram of clip
81     spect = zeros(s,length(tstep));
82     for k=1:length(tstep)
83         g=exp(-a*(t-tstep(k)).^2);
84         Sg=g'.*dat_3b(:,j); % apply new filter to the current 5sec
            clip
85         Sgt=fft(Sg); % transform to frequency domain
86         spect(:,k) = fftshift(abs(Sgt));
87     end
88     spect = spect((s/2-4000):(s/2+4000),:);
89     spect_array(:,j) = spect;
90     sp_3b(:,j) = reshape(spect,[8001*length(tstep),1]);
91 end
92
93
94 figure(1)
95 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
96 shading interp
97 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
98 set(gca,'FontSize',11)
99 colormap(hot)
100

```



```

101
102 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_03/test_03b-sp.mat sp_3b
103
104
105 %% Test 3 a
106
107 close all; clear all; clc;
108
109 load test_03/test_03a.mat
110
111
112 % set the time and frequency domains
113
114 size = size(dat_3a);
115 n=size(2); % number of samples
116 s = size(1); % length of samples
117 L=s/(44100/2); % time domain
118 t2=linspace(0,L,s+1);
119 t=t2(1:s);
120 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
121 ks_Hz=fftshift(k_Hz);
122 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
123
124 a = 2e4;
125 tstep=0:0.3:L;
126
127
128 % array of spectrogram data
129 sp_3a = zeros(8001*length(tstep),n);
130 spect_array = zeros(8001,length(tstep),n);
131
132 for j=1:n
133     % spectrogram of clip
134     spect = zeros(s,length(tstep));
135     for k=1:length(tstep)
136         g=exp(-a*(t-tstep(k)).^2);
137         Sg=g'.*dat_3a(:,j); % apply new filter to the current 5sec
            clip
138         Sgt=fft(Sg); % transform to frequency domain
139         spect(:,k) = fftshift(abs(Sgt));
140     end
141     spect = spect((s/2-4000):(s/2+4000),:);
142     spect_array(:,j) = spect;
143     sp_3a(:,j) = reshape(spect,[8001*length(tstep),1]);
144 end
145
146
147 figure(1)
148 pcolor(tstep,ks_Hz,spect_array(:, :, 60)),
149 shading interp
150 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
151 set(gca,'FontSize',11)
152 colormap(hot)

```

```

153
154 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
      HW_04/test_03/test_03a-sp.mat sp_3a
155
156
157 %% Test 2 c
158
159 close all; clear all; clc;
160
161 load test_02/test_02c.mat
162
163
164 % set the time and frequency domains
165
166 size = size(dat_2c);
167 n=size(2); % number of samples
168 s = size(1); % length of samples
169 L=s/(44100/2); % time domain
170 t2=linspace(0,L,s+1);
171 t=t2(1:s);
172 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
173 ks_Hz=fftshift(k_Hz);
174 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
175
176 a = 2e4;
177 tstep=0:0.3:L;
178
179
180 % array of spectrogram data
181 sp_2c = zeros(8001*length(tstep),n);
182 spect_array = zeros(8001,length(tstep),n);
183
184 for j=1:n
185     % spectrogram of clip
186     spect = zeros(s,length(tstep));
187     for k=1:length(tstep)
188         g=exp(-a*(t-tstep(k)).^2);
189         Sg=g'.* dat_2c(:,j); % apply new filter to the current 5sec
            clip
190         Sgt=fft(Sg); % transform to frequency domain
191         spect(:,k) = fftshift(abs(Sgt));
192     end
193     spect = spect((s/2-4000):(s/2+4000),:);
194     spect_array(:,:,j) = spect;
195     sp_2c(:,j) = reshape(spect,[8001*length(tstep),1]);
196 end
197
198
199 figure(1)
200 pcolor(tstep,ks_Hz,spect_array(:,:,20)),
201 shading interp
202 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
203 set(gca,'FontSize',11)
204 colormap(hot)

```

```

205
206 save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/HW_04
    /test_02/test_02c-sp.mat sp_2c
207 %% Test 2 b
208
209 close all; clear all; clc;
210
211 load test_02/test_02b.mat
212
213
214 % set the time and frequency domains
215
216 size = size(dat_2b);
217 n=size(2); % number of samples
218 s = size(1); % length of samples
219 L=s/(44100/2); % time domain
220 t2=linspace(0,L,s+1);
221 t=t2(1:s);
222 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
223 ks_Hz=fftshift(k_Hz);
224 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
225
226 a = 2e4;
227 tstep=0:0.3:L;
228
229
230 % array of spectrogram data
231 sp_2b = zeros(8001*length(tstep),n);
232 spect_array = zeros(8001,length(tstep),n);
233
234 for j=1:n
235     % spectrogram of clip
236     spect = zeros(s,length(tstep));
237     for k=1:length(tstep)
238         g=exp(-a*(t-tstep(k)).^2);
239         Sg=g'.*dat_2b(:,j); % apply new filter to the current 5sec
            clip
240         Sgt=fft(Sg); % transform to frequency domain
241         spect(:,k) = fftshift(abs(Sgt));
242     end
243     spect = spect((s/2-4000):(s/2+4000),:);
244     spect_array(:,j) = spect;
245     sp_2b(:,j) = reshape(spect,[8001*length(tstep),1]);
246 end
247
248
249 figure(1)
250 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
251 shading interp
252 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
253 set(gca,'FontSize',11)
254 colormap(hot)
255
256 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/

```

```

HW_04/test_02/test_02b-sp.mat sp-2b
257
258 %% Test 2 a
259
260 close all; clear all; clc;
261
262 load test_02/test_02a.mat
263
264
265 % set the time and frequency domains
266
267 size = size(dat_2a);
268 n=size(2); % number of samples
269 s = size(1); % length of samples
270 L=s/(44100/2); % time domain
271 t2=linspace(0,L,s+1);
272 t=t2(1:s);
273 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
274 ks_Hz=fftshift(k_Hz);
275 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
276
277 a = 2e4;
278 tstep=0:0.3:L;
279
280
281 % array of spectrogram data
282 sp_2a = zeros(8001*length(tstep),n);
283 spect_array = zeros(8001,length(tstep),n);
284
285 for j=1:n
286     % spectrogram of clip
287     spect = zeros(s,length(tstep));
288     for k=1:length(tstep)
289         g=exp(-a*(t-tstep(k)).^2);
290         Sg=g'.*dat_2a(:,j); % apply new filter to the current 5sec
                clip
291         Sgt=fft(Sg); % transform to frequency domain
292         spect(:,k) = fftshift(abs(Sgt));
293     end
294     spect = spect((s/2-4000):(s/2+4000),:);
295     spect_array(:,j) = spect;
296     sp_2a(:,j) = reshape(spect,[8001*length(tstep),1]);
297 end
298
299
300 figure(1)
301 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
302 shading interp
303 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
304 set(gca,'FontSize',11)
305 colormap(hot)
306
307 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_02/test_02a-sp.mat sp-2a

```

```

308
309
310 %% Test 1 c
311
312 close all; clear all; clc;
313
314 load test_01/test_01c.mat
315
316
317 % set the time and frequency domains
318
319 size = size(dat_1c);
320 n=size(2); % number of samples
321 s = size(1); % length of samples
322 L=s/(44100/2); % time domain
323 t2=linspace(0,L,s+1);
324 t=t2(1:s);
325 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
326 ks_Hz=fftshift(k_Hz);
327 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
328
329 a = 2e4;
330 tstep=0:0.3:L;
331
332
333 % array of spectrogram data
334 sp_1c = zeros(8001*length(tstep),n);
335 spect_array = zeros(8001,length(tstep),n);
336
337 for j=1:n
338     % spectrogram of clip
339     spect = zeros(s,length(tstep));
340     for k=1:length(tstep)
341         g=exp(-a*(t-tstep(k)).^2);
342         Sg=g'.*dat_1c(:,j); % apply new filter to the current 5sec
343         % clip
344         Sgt=fft(Sg); % transform to frequency domain
345         spect(:,k) = fftshift(abs(Sgt));
346     end
347     spect = spect((s/2-4000):(s/2+4000),:);
348     spect_array(:,j) = spect;
349     sp_1c(:,j) = reshape(spect,[8001*length(tstep),1]);
350 end
351
352 figure(1)
353 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
354 shading interp
355 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
356 set(gca,'FontSize',11)
357 colormap(hot)
358
359 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_01/test_01c-sp.mat sp_1c

```

```

360
361 %% Test 1 b
362
363 close all; clear all; clc;
364 load test_01/test_01b.mat
365
366
367 % set the time and frequency domains
368
369 size = size(dat_1b);
370 n=size(2); % number of samples
371 s = size(1); % length of samples
372 L=s/(44100/2); % time domain
373 t2=linspace(0,L,s+1);
374 t=t2(1:s);
375 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
376 ks_Hz=fftshift(k_Hz);
377 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
378
379 a = 2e4;
380 tstep=0:0.3:L;
381
382
383 % array of spectrogram data
384 sp_1b = zeros(8001*length(tstep),n);
385 spect_array = zeros(8001,length(tstep),n);
386
387 for j=1:n
388     % spectrogram of clip
389     spect = zeros(s,length(tstep));
390     for k=1:length(tstep)
391         g=exp(-a*(t-tstep(k)).^2);
392         Sg=g'.*dat_1b(:,j); % apply new filter to the current 5sec
393         % clip
394         Sgt=fft(Sg); % transform to frequency domain
395         spect(:,k) = fftshift(abs(Sgt));
396     end
397     spect = spect((s/2-4000):(s/2+4000),:);
398     spect_array(:,j) = spect;
399     sp_1b(:,j) = reshape(spect,[8001*length(tstep),1]);
400 end
401
402 figure(1)
403 pcolor(tstep,ks_Hz,spect_array(:, :, 20)),
404 shading interp
405 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
406 set(gca,'FontSize',11)
407 colormap(hot)
408
409 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
410     HW_04/test_01/test_01b-sp.mat sp_1b
411
412 %% Test 1 a

```

```

412
413 close all; clear all; clc;
414
415 load test_01/test_01a.mat
416
417 % set the time and frequency domains
418
419 size = size(dat_1a);
420 n=size(2); % number of samples
421 s = size(1); % length of samples
422 L=s/(44100/2); % time domain
423 t2=linspace(0,L,s+1);
424 t=t2(1:s);
425 k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
426 ks_Hz=fftshift(k_Hz);
427 ks_Hz = ks_Hz((s/2-4000):(s/2+4000));
428
429 a = 2e4;
430 tstep=0:0.3:L;
431
432
433 % array of spectrogram data
434 sp_1a = zeros(8001*length(tstep),n);
435 spect_array = zeros(8001,length(tstep),n);
436
437 for j=1:n
438     % spectrogram of clip
439     spect = zeros(s,length(tstep));
440     for k=1:length(tstep)
441         g=exp(-a*(t-tstep(k)).^2);
442         Sg=g'.* dat_1a(:,j); % apply new filter to the current 5sec
443         % clip
444         Sgt=fft(Sg); % transform to frequency domain
445         spect(:,k) = fftshift(abs(Sgt));
446     end
447     spect = spect((s/2-4000):(s/2+4000),:);
448     spect_array(:,:,j) = spect;
449     sp_1a(:,j) = reshape(spect,[8001*length(tstep),1]);
450 end
451 %spect = reshape(sp_1a(:,1),[length(ks_Hz),length(tstep)]);
452
453 figure(1)
454 pcolor(tstep,ks_Hz,spect_array(:,:,20)),
455 shading interp
456 title('Unfiltered: a = 1e03, inc = 0.0446','FontSize',16)
457 set(gca,'FontSize',11)
458 colormap(hot)
459
460 % save /Users/christinasmith/Desktop/AMATH482.nosync/Homework/
    HW_04/test_01/test_01a-sp.mat sp_1a

```

HW\_04\_combined.m

```

1  % Audio classifier
2
3  %% Load in spectrogram files
4
5  clear; close all; clc
6
7  load test_01/test_01a_sp.mat
8  load test_01/test_01b_sp.mat
9  load test_01/test_01c_sp.mat
10
11 load test_02/test_02a_sp.mat
12 load test_02/test_02b_sp.mat
13 load test_02/test_02c_sp.mat
14
15 load test_03/test_03a_sp.mat
16 load test_03/test_03b_sp.mat
17 load test_03/test_03c_sp.mat
18
19 %% Train classifier
20
21 feature = 50;
22 size_train = 25;
23 size_test = 15;
24
25 feature = 80;
26 size_train = 55;
27 size_test = 15;
28
29 % train_a = sp_1a(:,1:size_train);
30 % train_b = sp_1b(:,1:size_train);
31 % train_c = sp_1c(:,1:size_train);
32 %
33 % train_a = sp_2a(:,1:size_train);
34 % train_b = sp_2b(:,1:size_train);
35 % train_c = sp_2c(:,1:size_train);
36
37 train_a = sp_3a(:,1:size_train);
38 train_b = sp_3b(:,1:size_train);
39 train_c = sp_3c(:,1:size_train);
40
41
42
43
44 [U,S,V,thresholds,d_arr,w_arr,rank,proj_w1, proj_w2] = trainer(
    train_3a,train_3b,train_3c,feature);
45
46 % compute the mean position in the w1/w2 plane
47 class_1 = [mean(proj_w1(1,:)),mean(proj_w2(1,:))];
48 class_2 = [mean(proj_w1(2,:)),mean(proj_w2(2,:))];
49 class_3 = [mean(proj_w1(3,:)),mean(proj_w2(3,:))];
50
51 means = [class_1; class_2; class_3];
52
53 %% Test 3

```



```

54
55 dim = size(sp_3a);
56 n = dim(2);
57
58 % % % test the different classes
59 test_a = sp_3a(:,size_train+1:n);
60 test_b = sp_3b(:,size_train+1:n);
61 test_c = sp_3c(:,size_train+1:n);
62
63 % project the test data on the principal components & w vectors
64 pval_a = w_arr*U'*test_a;
65 pval_b = w_arr*U'*test_b;
66 pval_c = w_arr*U'*test_c;
67
68
69 % class_a1 = classify_1D(pval_a(1,:), thresholds(1,:), rank);
70 % class_b1 = classify_1D(pval_b(1,:), thresholds(1,:), rank);
71 % class_c1 = classify_1D(pval_c(1,:), thresholds(1,:), rank);
72 %
73 % tc_a = sum(class_a1 == 1);
74 % tt_a = length(class_a1);
75 % pc_a = tc_a/tt_a;
76 %
77 % tc_b = sum(class_b1 == 2);
78 % tt_b = length(class_b1);
79 % pc_b = tc_b/tt_b;
80 %
81 % tc_c = sum(class_c1 == 3);
82 % tt_c = length(class_c1);
83 % pc_c = tc_c/tt_c;
84
85 class_a2 = classify_2D(pval_a, means);
86 class_b2 = classify_2D(pval_b, means);
87 class_c2 = classify_2D(pval_c, means);
88
89 tc_a = sum(class_a2 == 1);
90 tt_a = length(class_a2);
91 pc_a = tc_a/tt_a;
92
93 tc_b = sum(class_b2 == 2);
94 tt_b = length(class_b2);
95 pc_b = tc_b/tt_b;
96
97 tc_c = sum(class_c2 == 3);
98 tt_c = length(class_c2);
99 pc_c = tc_c/tt_c;
100
101 %% Plot the test and training data in 1D
102
103 figure(1)
104 subplot(1,2,1)
105 plot(proj_w1(1,:), ones(size_train), 'ob', 'Linewidth', 2)
106 hold on
107 plot(proj_w1(2,:), ones(size_train)*2, 'or', 'Linewidth', 2)

```

```

108 hold on
109 plot(proj_w1(3,:),ones(size_train)*3,'og','Linewidth',2)
110 ylim([0.5 3.5])
111 plot([thresholds(1,1);thresholds(1,1)],get(gca,'ylim'),'r')
112 hold on
113 plot([thresholds(1,2);thresholds(1,2)],get(gca,'ylim'),'r')
114 title('Test 3: training data','FontSize',16)
115 LH(1) = plot(nan,nan,'ob');
116 L{1} = 'CLASS A';
117 LH(2) = plot(nan,nan,'or');
118 L{2} = 'CLASS B';
119 LH(3) = plot(nan,nan,'og');
120 L{3} = 'CLASS C';
121 legend(LH,L,'FontSize',12,'Location','northeast');
122 xlabel('w1 values','FontSize',12,'Fontweight','bold');
123 yticks([1 2 3])
124 yticklabels({'Class A','Class B','Class C'})
125 hold off
126
127 % Plot the test data 1D
128 subplot(1,2,2)
129 plot(pval_a(1,:),ones(length(pval_a(1,:))), 'ob','Linewidth',2)
130 hold on
131 plot(pval_b(1,:),ones(length(pval_b(1,:)))*2,'or','Linewidth',2)
132 hold on
133 plot(pval_c(1,:),ones(length(pval_c(1,:)))*3,'og','Linewidth',2)
134 hold on
135 ylim([0.5 3.5])
136 plot([thresholds(1,1);thresholds(1,1)],get(gca,'ylim'),'r')
137 hold on
138 plot([thresholds(1,2);thresholds(1,2)],get(gca,'ylim'),'r')
139 title('Test 3: test data','FontSize',16)
140 LH(1) = plot(nan,nan,'ob');
141 L{1} = 'CLASS A';
142 LH(2) = plot(nan,nan,'or');
143 L{2} = 'CLASS B';
144 LH(3) = plot(nan,nan,'og');
145 L{3} = 'CLASS C';
146 legend(LH,L,'FontSize',12,'Location','northeast');
147 xlabel('w1 values','FontSize',12,'Fontweight','bold');
148 yticks([1 2 3])
149 yticklabels({'Class A','Class B','Class C'})
150 hold off
151
152 %% Plot the test and training data in 2D
153
154
155 figure(2)
156 subplot(1,2,1)
157 plot(proj_w1(1,:),proj_w2(1,:), 'bo')
158 hold on;
159 plot(class_1(1),class_1(2), 'o','MarkerFaceColor','b')
160 hold on;
161 plot(proj_w1(2,:),proj_w2(2,:), 'ro')

```

```

162 hold on;
163 plot(class_2(1),class_2(2), 'o','MarkerFaceColor','r')
164 hold on;
165 plot(proj_w1(3,:),proj_w2(3,:), 'go')
166 hold on;
167 plot(class_3(1),class_3(2), 'o','MarkerFaceColor','g')
168 hold on;
169 title('Test 3: training data','FontSize',16)
170 xlabel('w1 values','FontSize',12,'Fontweight','bold');
171 ylabel('w2 values','FontSize',12,'Fontweight','bold');
172 LH(1) = plot(nan, nan, 'ob');
173 L{1} = 'CLASS A';
174 LH(2) = plot(nan, nan, 'or');
175 L{2} = 'CLASS B';
176 LH(3) = plot(nan, nan, 'og');
177 L{3} = 'CLASS C';
178 legend(LH, L, 'FontSize',12, 'Location','southeast');
179 hold off
180
181 % Plot test data 2D
182
183 subplot(1,2,2)
184 plot(means(1,1),means(1,2), 'o','MarkerFaceColor','b')
185 hold on;
186 plot(means(2,1),means(2,2), 'o','MarkerFaceColor','r')
187 hold on;
188 plot(means(3,1),means(3,2), 'o','MarkerFaceColor','g')
189 hold on;
190 plot(pval_a(1,:),pval_a(2,:), 'bo')
191 hold on;
192 plot(pval_b(1,:),pval_b(2,:), 'ro')
193 hold on;
194 plot(pval_c(1,:),pval_c(2,:), 'go')
195 title('Test 3: test data','FontSize',16)
196 xlabel('w1 values','FontSize',12,'Fontweight','bold');
197 ylabel('w2 values','FontSize',12,'Fontweight','bold');
198 LH(1) = plot(nan, nan, 'ob');
199 L{1} = 'CLASS A';
200 LH(2) = plot(nan, nan, 'or');
201 L{2} = 'CLASS B';
202 LH(3) = plot(nan, nan, 'og');
203 L{3} = 'CLASS C';
204 legend(LH, L, 'FontSize',12, 'Location','southeast');
205 hold off
206
207
208 %% Plot histograms of test data 1D
209
210 % bins = 1;
211 %
212 %% Plot histogram of test data with thresholds
213 % figure(3)
214 % subplot(1,3,1)
215 % histogram(pval_a(1,:),bins);

```

```

216 % xlim([-500 1500])
217 % hold on,
218 % plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
219 % hold on
220 % plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
221 % xlabel('class 2 range          class 1 range
        class 3 range',...
222 %         'FontSize', 10, 'Fontweight', 'bold')
223 % title('Test 1: class 1 data', 'FontSize', 14)
224 % hold off
225 %
226 % subplot(1,3,2)
227 % histogram(pval_b(1,:),bins);
228 % xlim([-500 1500])
229 % hold on,
230 % plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
231 % hold on
232 % plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
233 % xlabel('class 2 range          class 1 range
        class 3 range',...
234 %         'FontSize', 10, 'Fontweight', 'bold')
235 % title('Test 1: class 2 data', 'FontSize', 14)
236 % hold off
237 %
238 % subplot(1,3,3)
239 % histogram(pval_c(1,:),bins);
240 % xlim([-500 1500])
241 % hold on,
242 % plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
243 % hold on
244 % plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
245 % xlabel('class 2 range          class 1 range
        class 3 range',...
246 %         'FontSize', 10, 'Fontweight', 'bold')
247 % title('Test 1: class 3 data', 'FontSize', 14)
248 % hold off
249
250 %% Classify using 1D
251
252 % Classify using first projection vector
253
254 function classes = classify_1D(pval,threshold,rank)
255
256     dim = size(pval);
257     n = dim(2);
258     classes = zeros(1,n);
259
260     for j=1:n
261         if pval(j)<threshold(2) && pval(j)>threshold(1)
262             classes(j) = rank(2);
263         elseif pval(j)>threshold(2)
264             classes(j) = rank(3);
265         else
266             classes(j) = rank(1);

```

```

267         end
268     end
269 end
270
271
272 %% Classify 2D
273
274 % Classify using both projection vectors
275
276 function classes = classify_2D(pvals, means)
277
278     dim = size(pvals);
279     n = dim(2);
280     classes = zeros(1, n);
281
282     % for every spectrogram in the data
283     for j=1:n
284         A = [pvals(:, j)'; means(1, :)]';
285         B = [pvals(:, j)'; means(2, :)]';
286         C = [pvals(:, j)'; means(3, :)]';
287
288         % compute the minimum distance to a class mean
289         distances = [pdist(A, 'euclidean') ...
290                     pdist(B, 'euclidean') ...
291                     pdist(C, 'euclidean')];
292
293         [~, min_idx] = min(distances);
294         classes(j) = min_idx;
295     end
296
297 end
298
299
300 function [U, S, V, thresholds, d_arr, w_arr, rank, proj_w1, proj_w2] =
    trainer(s1, s2, s3, feature)
301
302     % a 800k by 30 matrix of the spectrograms from all 3 groups
303     X = [s1 s2 s3];
304
305     ns1 = size(s1, 2);
306     ns2 = size(s2, 2);
307     ns3 = size(s3, 2);
308
309     [U, S, V] = svd(X, 'econ');
310
311     songs = S*V'; % projection onto principal components
312
313     % choose how many singular values we want
314     U = U(:, 1:feature);
315
316     % separate the songs now that they are
317     % in the principal component basis
318     song_1 = songs(1:feature, 1:ns1);
319     song_2 = songs(1:feature, ns1+1:ns1+ns2);

```

```

320 song_3 = songs(1:feature, ns1+ns2+1:ns1+ns2+ns3);
321
322 ms_all = mean(songs(1:feature, :), 2);
323 ms1 = mean(song_1, 2);
324 ms2 = mean(song_2, 2);
325 ms3 = mean(song_3, 2);
326
327
328 Sw = 0; % within class variances
329 for k=1:ns1
330     Sw = Sw + (song_1(:,k)-ms1)*(song_1(:,k)-ms1)';
331 end
332 for k=1:ns2
333     Sw = Sw + (song_2(:,k)-ms2)*(song_2(:,k)-ms2)';
334 end
335 for k=1:ns3
336     Sw = Sw + (song_3(:,k)-ms3)*(song_3(:,k)-ms3)';
337 end
338
339 Sb = (ms1-ms_all)*(ms1-ms_all)'; % between class
340 Sb = Sb + (ms2-ms_all)*(ms2-ms_all)';
341 Sb = Sb + (ms3-ms_all)*(ms3-ms_all)';
342
343
344 % get the eigenvalue
345 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
346 max(max(D));
347
348 [d1, ind] = max(abs(diag(D)));
349
350 % eigenvector with max eigenvalue
351 w1 = V2(:, ind);
352 w1 = w1/norm(w1,2);
353
354 D(:, ind) = [];
355 [d2, ind] = max(abs(diag(D)));
356
357 d_arr = [d1, d2];
358
359 % eigenvector with 2nd max eigenvalue
360 w2 = V2(:, ind);
361 w2 = w2/norm(w2,2);
362
363 w_arr = [w1 w2]';
364
365 % project the song data onto w1
366 sort_s1a = w1'*song_1;
367 sort_s2a = w1'*song_2;
368 sort_s3a = w1'*song_3;
369
370 proj_w1 = [sort_s1a; sort_s2a; sort_s3a];
371
372 % project the song data onto w2
373 sort_s1b = w2'*song_1;

```

```

374     sort_s2b = w2'*song_2;
375     sort_s3b = w2'*song_3;
376
377     proj_w2 = [sort_s1b; sort_s2b; sort_s3b];
378
379
380     % figure out which means are 'neighbors'
381
382     mv_a = [mean(sort_s1a) mean(sort_s2a) mean(sort_s3a)];
383
384     for j=1:3
385         if mv_a(j) == max(mv_a)
386             % max_val = mv_a(j);
387             max_class_a = j;
388         elseif mv_a(j) == min(mv_a)
389             % min_val = mv_a(j);
390             min_class_a = j;
391         else
392             % med_val = mv_a(j);
393             med_class_a = j;
394         end
395     end
396
397     rank = [min_class_a med_class_a max_class_a];
398
399     vsm = [sort_s1a; sort_s2a; sort_s3a];
400
401     % sort all values from lowest to highest
402     sort_max_a = sort(vsm(max_class_a,:));
403     sort_med_a = sort(vsm(med_class_a,:));
404     sort_min_a = sort(vsm(min_class_a,:));
405
406     %     thresholds(1,1) = (mean(sort_med_a) - mean(sort_min_a))/2;
407     %     thresholds(1,2) = (mean(sort_max_a) - mean(sort_med_a))/2;
408     t1a = length(sort_max_a);
409     t2a = 1;
410     while (t1a > 0 && sort_max_a(t1a)>sort_med_a(t2a))
411         t1a = t1a-1;
412         t2a = t2a+1;
413     end
414     if t1a == 0
415         thresholds(1,2) = (sort_max_a(1)+sort_med_a(length(sort_max_a)
416             ))/2;
417     else
418         thresholds(1,2) = (sort_max_a(t1a)+sort_med_a(t2a))/2;
419     end
420
421     t1b = length(sort_med_a);
422     t2b = 1;
423     while (t1b > 0 && sort_med_a(t1b)>sort_min_a(t2b))
424         t1b = t1b-1;
425         t2b = t2b+1;
426     end

```

```

427
428     if t1b == 0
429         thresholds(1,1) = (sort_med_a(1)+sort_min_a(length(sort_med_a)
430             ))/2;
431     else
432         thresholds(1,1) = (sort_med_a(t1b)+sort_min_a(t2b))/2;
433     end
434     % figure out which means are 'neighbors'
435
436     mv_b = [mean(sort_s1b) mean(sort_s2b) mean(sort_s3b)];
437
438     for j=1:3
439         if mv_b(j) == max(mv_b)
440             % max_val = mv_a(j);
441             max_class_b = j;
442         elseif mv_b(j) == min(mv_b)
443             % min_val = mv_a(j);
444             min_class_b = j;
445         else
446             % med_val = mv_a(j);
447             med_class_b = j;
448         end
449     end
450
451     vsm = [sort_s1b; sort_s2b; sort_s3b];
452
453     % sort all values from lowest to highest
454     sort_max_b = sort(vsm(max_class_b,:));
455     sort_med_b = sort(vsm(med_class_b,:));
456     sort_min_b = sort(vsm(min_class_b,:));
457
458
459
460     %     thresholds(2,1) = (mean(sort_med_b) - mean(sort_min_b))/2;
461     %     thresholds(2,2) = (mean(sort_max_b) - mean(sort_med_b))/2;
462
463
464     t1a = length(sort_max_b);
465     t2a = 1;
466     while (t1a > 0 && sort_max_b(t1a)>sort_med_b(t2a))
467         t1a = t1a-1;
468         t2a = t2a+1;
469     end
470     if t1a == 0
471         thresholds(2,2) = (sort_max_b(1)+sort_med_b(length(sort_max_b)
472             ))/2;
473     else
474         thresholds(2,2) = (sort_max_b(t1a)+sort_med_b(t2a))/2;
475     end
476
477     t1b = length(sort_med_b);
478     t2b = 1;

```



```

479     while (t1b > 0 && sort_med_b(t1b)>sort_min_b(t2b))
480         t1b = t1b-1;
481         t2b = t2b+1;
482     end
483
484     if t1b == 0
485         thresholds(2,1) = (sort_med_b(1)+sort_min_b(length(sort_med_b)
486             ))/2;
487     else
488         thresholds(2,1) = (sort_med_b(t1b)+sort_min_b(t2b))/2;
489     end
490 end

```