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:

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

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:

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

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

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \tag{3}$$

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 **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 **w1** and **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.

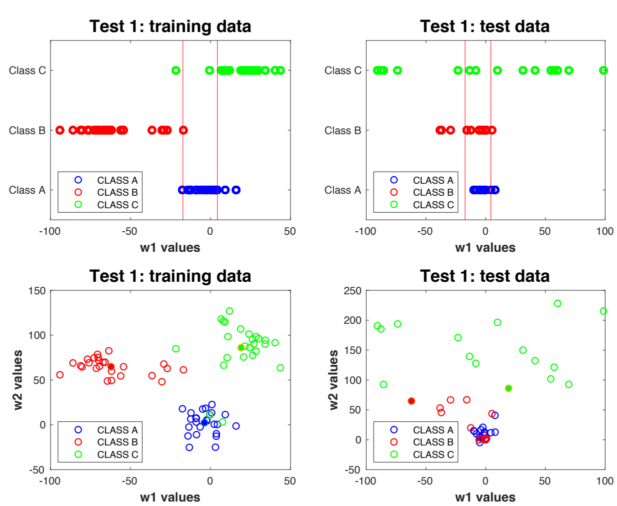

Figure 1: Test 1: Projections of Training and Test Data

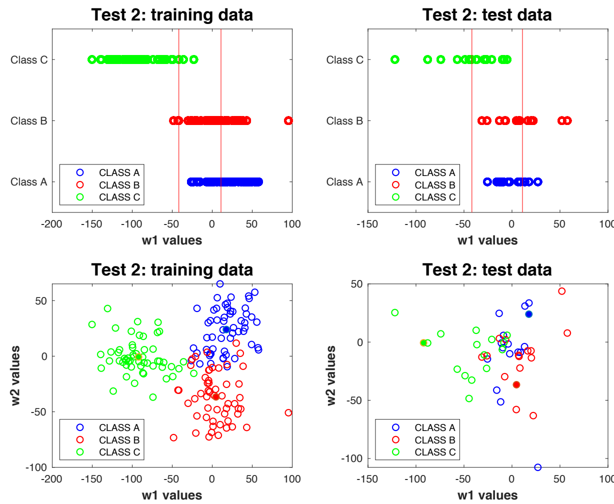

Figure 2: Test 2: Projections of Training and Test Data

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 **w1** 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 **w1** projection, Class B was the middle range, and Class A was the upper range. The threshold values were -41.8 and 10.9.

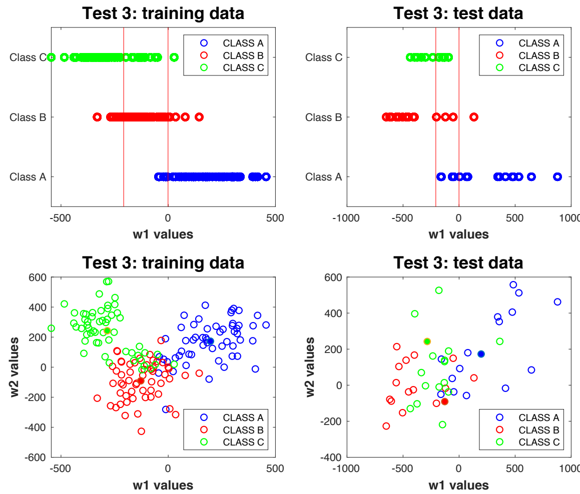

Figure 3: Test 3: Projections of Training and Test Data

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 that 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 **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

```
% %% Test 3 c
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_3c = zeros(s,70); % 40 samples total (10/song)
%
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_03/rock/all_songs/0',num2str(i),'.mp3');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_3c(:,count+1) = y_resampled(1:s);   % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/rock/all_clips/0',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_3c(:,count+1),Fs/2);
%         else
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/rock/all_clips/00',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_3c(:,count+1),Fs/2);
%         end
%         count = count+1;
%     end
% end
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/test_03c.mat dat_3c

% %% Test 3 b
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_3b = zeros(s,70); % 70 samples total (10/song)
%
```

```
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_03/country/all_songs/0',num2str(i),'.mp3');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_3b(:,count+1) = y_resampled(1:s);   % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/country/all_clips/0',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_3b(:,count+1),Fs/2);
%         else
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/country/all_clips/00',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_3b(:,count+1),Fs/2);
%         end
%         count = count+1;
%     end
% end
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/test_03b.mat dat_3b
% %
% %% Test 3 a
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_3a = zeros(s,40); % 40 samples total (10/song)
%
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_03/chill_beats/all_songs/0',num2str(i),'.mp3');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
```

```
%       for j=1:10
%           dat_3a(:,count+1) = y_resampled(1:s);    % new 5 second clip
%           last = size(y_resampled);
%           y_resampled = y_resampled(s+1:last(2));
%           if count > 9
%               wavname =...
%                   strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/chill_beats/all_clips/0',...
%               num2str(count),'.wav');
%               audiowrite(wavname,dat_3a(:,count+1),Fs/2);
%           else
%               wavname =...
%                   strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/chill_beats/all_clips/00',...
%               num2str(count),'.wav');
%               audiowrite(wavname,dat_3a(:,count+1),Fs/2);
%           end
%           count = count+1;
%       end
% end
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/test_03a.mat dat_3a
%
%
% %% Test 2c

% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_2c = zeros(s,70); % 70 samples total (10/song)
%
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_02/sam_cooke/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_2c(:,count+1) = y_resampled(1:s);    % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/sam_cooke/all_clips/0',...
```

```
%                 num2str(count),'.wav');
%                 audiowrite(wavname,dat_2c(:,count+1),Fs/2);
%             else
%                 wavname =...
%                     strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/sam_cooke/all_clips/00',...
%                 num2str(count),'.wav');
%                 audiowrite(wavname,dat_2c(:,count+1),Fs/2);
%             end
%             count = count+1;
%         end
% end
%
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/test_02c.mat dat_2c
%
%% Test 2 b

% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_2b = zeros(s,70); % 70 samples total (10/song)
%
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_02/otis_redding/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%             dat_2b(:,count+1) = y_resampled(1:s);    % new 5 second clip
%             last = size(y_resampled);
%             y_resampled = y_resampled(s+1:last(2));
%             if count > 9
%                 wavname =...
%                     strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/otis_redding/all_clips/0',...
%                 num2str(count),'.wav');
%                 audiowrite(wavname,dat_2b(:,count+1),Fs/2);
%             else
%                 wavname =...
%                     strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/otis_redding/all_clips/00',...
%                 num2str(count),'.wav');
%                 audiowrite(wavname,dat_2b(:,count+1),Fs/2);
%             end
```

```
%             count = count+1;
%         end
% end
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/test_02b.mat dat_2b


% %% Test 2 a
%
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_2a = zeros(s,40); % 40 samples total (10/song)
%
% count = 0;
% for i=1:7
%     % go through each song
%     filename = strcat('test_02/al_green/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_2a(:,count+1) = y_resampled(1:s);   % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/al_green/all_clips/0',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_2a(:,count+1),Fs/2);
%         else
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/al_green/all_clips/00',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_2a(:,count+1),Fs/2);
%         end
%         count = count+1;
%     end
% end
%
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/test_02a.mat dat_2a
%
%
```

```
% %% Test 1 c
%
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_1c = zeros(s,40); % 40 samples total (10/song)
%
% count = 0;
% for i=1:4
%     % go through each song
%     filename = strcat('test_01/air/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%           dat_1c(:,count+1) = y_resampled(1:s);    % new 5 second clip
%           last = size(y_resampled);
%           y_resampled = y_resampled(s+1:last(2));
%           if count > 9
%               wavname =...
%                   strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/air/all_clips/0',...
%               num2str(count),'.wav');
%               audiowrite(wavname,dat_1c(:,count+1),Fs/2);
%           else
%               wavname =...
%                   strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/air/all_clips/00',...
%               num2str(count),'.wav');
%               audiowrite(wavname,dat_1c(:,count+1),Fs/2);
%           end
%           count = count+1;
%     end
% end
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01c.mat dat_1c
%
% %% Test 1 b
% close all; clear all; clc;
%
% s = (50000)*2;
%
% dat_1b = zeros(s,40); % 40 samples total (10/song)
%
% count = 0;
% for i=1:4
%     % go through each song
```

```
%     filename = strcat('test_01/nina/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_1b(:,count+1) = y_resampled(1:s);   % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/nina/all_clips/0',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_1b(:,count+1),Fs/2);
%         else
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/nina/all_clips/00',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_1b(:,count+1),Fs/2);
%         end
%         count = count+1;
%
%     end
% end
%
%
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01b.mat dat_1b
%
% %% Test 1 a
%
%
% close all; clear all; clc;
%
% s = (50000)*2;
%
%
% dat_1a = zeros(s,40); % 40 samples total (10/song)
%
% count = 0;
% for i=1:3
%     % Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/air/all_songs/01.wav
%     % go through each song
%     filename = strcat('test_01/rubblebucket/all_songs/0',num2str(i),'.wav');
%     [y,Fs] = audioread(filename);
%     y_mono = y(:,1);  % mono
```

```
%     b = 1:2:length(y_mono); % resampling vector
%     y_resampled = y_mono(b)';  % transpose the y vector
%
%     % get 10 samples of each song
%     for j=1:10
%         dat_1a(:,count+1) = y_resampled(1:s);   % new 5 second clip
%         last = size(y_resampled);
%         y_resampled = y_resampled(s+1:last(2));
%         if count > 9
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/rubblebucket/all_clips/0',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_1a(:,count+1),Fs/2);
%         else
%             wavname =...
%                 strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/rubblebucket/all_clips/00',...
%             num2str(count),'.wav');
%             audiowrite(wavname,dat_1a(:,count+1),Fs/2);
%         end
%         count = count+1;
%     end
% end
%
%
% filename = strcat('test_01/rubblebucket/all_songs/la.wav');
% [y,Fs] = audioread(filename);
% y_mono = y(:,1);  % mono
% b = 1:2:length(y_mono); % resampling vector
% y_resampled = y_mono(b)';  % transpose the y vector
%
% for j=1:9
%     dat_1a(:,count+1) = y_resampled(1:s);   % new 5 second clip
%     last = size(y_resampled);
%     y_resampled = y_resampled(s+1:last(2));
%     if count > 9
%         wavname =...
%             strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/rubblebucket/all_clips/0',...
%         num2str(count),'.wav');
%         audiowrite(wavname,dat_1a(:,count+1),Fs/2);
%     else
%         wavname =...
%             strcat('/Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/rubblebucket/all_clips/00',...
%         num2str(count),'.wav');
%         audiowrite(wavname,dat_1a(:,count+1),Fs/2);
%     end
%     count = count+1;
% end
%
%
```

```
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01a.mat dat_1a
```

create_spectrograms.m

```
%% Test 3 c

close all; clear all; clc;

load test_03/test_03c.mat


% set the time and frequency domains

size = size(dat_3c);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);   % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_3c = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_3c(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_3c(:,j) = reshape(spect,[8001*length(tstep),1]);
end

figure(1)
subplot(1,1,1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Home-made a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)
```

```
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/test_03c_sp.mat sp_3c

%% Test 3 b

close all; clear all; clc;

load test_03/test_03b.mat


% set the time and frequency domains

size = size(dat_3b);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);   % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_3b = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_3b(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_3b(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

```

```
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_03/test_03b_sp.mat sp_3b


%% Test 3 a

close all; clear all; clc;

load test_03/test_03a.mat


% set the time and frequency domains

size = size(dat_3a);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);   % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_3a = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_3a(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_3a(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,60)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)
```

```
% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/
    HW_04/test_03/test_03a_sp.mat sp_3a


%% Test 2 c

close all; clear all; clc;

load test_02/test_02c.mat


% set the time and frequency domains

size = size(dat_2c);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);   % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_2c = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_2c(:,j); % apply new filter to the current 5sec
            clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_2c(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)
```

```

save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_02/test_02c_sp.mat sp_2c
%% Test 2 b

close all; clear all; clc;

load test_02/test_02b.mat


% set the time and frequency domains

size = size(dat_2b);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);  % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_2b = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_2b(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_2b(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/
```

```
    HW_04/test_02/test_02b_sp.mat sp_2b

%% Test 2 a

close all; clear all; clc;

load test_02/test_02a.mat


% set the time and frequency domains

size = size(dat_2a);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);  % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_2a = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_2a(:,j); % apply new filter to the current 5sec
            clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_2a(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/
    HW_04/test_02/test_02a_sp.mat sp_2a
```

```
%% Test 1 c

close all; clear all; clc;

load test_01/test_01c.mat


% set the time and frequency domains

size = size(dat_1c);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);  % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_1c = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_1c(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_1c(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01c_sp.mat sp_1c
```

```
%% Test 1 b

close all; clear all; clc;
load test_01/test_01b.mat


% set the time and frequency domains

size = size(dat_1b);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);  % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_1b = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_1b(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_1b(:,j) = reshape(spect,[8001*length(tstep),1]);
end


figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01b_sp.mat sp_1b

%% Test 1 a
```

```
close all; clear all; clc;

load test_01/test_01a.mat

% set the time and frequency domains

size = size(dat_1a);
n=size(2);    % number of samples
s = size(1); % length of samples
L=s/(44100/2);  % time domain
t2=linspace(0,L,s+1);
t=t2(1:s);
k_Hz= (1/(2*L))*[0:(s/2-1) -s/2:-1];
ks_Hz=fftshift(k_Hz);
ks_Hz = ks_Hz((s/2-4000):(s/2+4000));

a = 2e4;
tstep=0:0.3:L;


% array of spectrogram data
sp_1a = zeros(8001*length(tstep),n);
spect_array = zeros(8001,length(tstep),n);

for j=1:n
    % spectrogram of clip
    spect = zeros(s,length(tstep));
    for k=1:length(tstep)
        g=exp(-a*(t-tstep(k)).^2);
        Sg=g'.*dat_1a(:,j); % apply new filter to the current 5sec clip
        Sgt=fft(Sg); % transform to frequency domain
        spect(:,k) = fftshift(abs(Sgt));
    end
    spect = spect((s/2-4000):(s/2+4000),:);
    spect_array(:,:,j) = spect;
    sp_1a(:,j) = reshape(spect,[8001*length(tstep),1]);
end

%spect = reshape(sp_1a(:,1),[length(ks_Hz),length(tstep)]);

figure(1)
pcolor(tstep,ks_Hz,spect_array(:,:,20)),
shading interp
title('Unfiltered: a = 1e03, inc = 0.0446','Fontsize',16)
set(gca,'Fontsize',11)
colormap(hot)

% save /Users/christinasmith/Desktop/AMATH_482.nosync/Homework/HW_04/test_01/test_01a_sp.mat sp_1a
```

HW_04_combined.m

```
% Audio classifier

%% Load in spectrogram files

clear; close all; clc

load test_01/test_01a_sp.mat
load test_01/test_01b_sp.mat
load test_01/test_01c_sp.mat

load test_02/test_02a_sp.mat
load test_02/test_02b_sp.mat
load test_02/test_02c_sp.mat

load test_03/test_03a_sp.mat
load test_03/test_03b_sp.mat
load test_03/test_03c_sp.mat

%% Train classifier

feature = 50;
size_train = 25;
size_test = 15;

feature = 80;
size_train = 55;
size_test = 15;

% train_a = sp_1a(:,1:size_train);
% train_b = sp_1b(:,1:size_train);
% train_c = sp_1c(:,1:size_train);
%
% train_a = sp_2a(:,1:size_train);
% train_b = sp_2b(:,1:size_train);
% train_c = sp_2c(:,1:size_train);

train_a = sp_3a(:,1:size_train);
train_b = sp_3b(:,1:size_train);
train_c = sp_3c(:,1:size_train);




[U,S,V,thresholds,d_arr,w_arr,rank,proj_w1, proj_w2] = trainer(
    train_3a,train_3b,train_3c,feature);

% compute the mean position in the w1/w2 plane
class_1 = [mean(proj_w1(1,:)),mean(proj_w2(1,:))];
class_2 = [mean(proj_w1(2,:)),mean(proj_w2(2,:))];
class_3 = [mean(proj_w1(3,:)),mean(proj_w2(3,:))];

means = [class_1; class_2; class_3];

%% Test 3
```

```
dim = size(sp_3a);
n = dim(2);

%%% test the different classes
test_a = sp_3a(:,size_train+1:n);
test_b = sp_3b(:,size_train+1:n);
test_c = sp_3c(:,size_train+1:n);

% project the test data on the principal components & w vectors
pval_a = w_arr*U'*test_a;
pval_b = w_arr*U'*test_b;
pval_c = w_arr*U'*test_c;


% class_a1 = classify_1D(pval_a(1,:),thresholds(1,:), rank);
% class_b1 = classify_1D(pval_b(1,:),thresholds(1,:), rank);
% class_c1 = classify_1D(pval_c(1,:),thresholds(1,:), rank);
%
% tc_a = sum(class_a1 == 1);
% tt_a = length(class_a1);
% pc_a = tc_a/tt_a;
%
% tc_b = sum(class_b1 == 2);
% tt_b = length(class_b1);
% pc_b = tc_b/tt_b;
%
% tc_c = sum(class_c1 == 3);
% tt_c = length(class_c1);
% pc_c = tc_c/tt_c;

class_a2 = classify_2D(pval_a,means);
class_b2 = classify_2D(pval_b,means);
class_c2 = classify_2D(pval_c,means);

tc_a = sum(class_a2 == 1);
tt_a = length(class_a2);
pc_a = tc_a/tt_a;

tc_b = sum(class_b2 == 2);
tt_b = length(class_b2);
pc_b = tc_b/tt_b;

tc_c = sum(class_c2 == 3);
tt_c = length(class_c2);
pc_c = tc_c/tt_c;

%% Plot the test and training data in 1D

figure(1)
subplot(1,2,1)
plot(proj_w1(1,:),ones(size_train),'ob','Linewidth',2)
hold on
plot(proj_w1(2,:),ones(size_train)*2,'or','Linewidth',2)
```

```
hold on
plot(proj_w1(3,:),ones(size_train)*3,'og','Linewidth',2)
ylim([0.5 3.5])
plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
hold on
plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
title('Test 3: training data', 'Fontsize', 16)
LH(1) = plot(nan, nan, 'ob');
L{1} = 'CLASS A';
LH(2) = plot(nan, nan, 'or');
L{2} = 'CLASS B';
LH(3) = plot(nan, nan, 'og');
L{3} = 'CLASS C';
legend(LH, L, 'FontSize',12, 'Location', 'northeast');
xlabel('w1 values','FontSize',12, 'Fontweight', 'bold');
yticks([1 2 3])
yticklabels({'Class A','Class B','Class C'})
hold off

% Plot the test data 1D
subplot(1,2,2)
plot(pval_a(1,:),ones(length(pval_a(1,:))),'ob','Linewidth',2)
hold on
plot(pval_b(1,:),ones(length(pval_b(1,:)))*2,'or','Linewidth',2)
hold on
plot(pval_c(1,:),ones(length(pval_c(1,:)))*3,'og','Linewidth',2)
hold on
ylim([0.5 3.5])
plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
hold on
plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
title('Test 3: test data', 'Fontsize', 16)
LH(1) = plot(nan, nan, 'ob');
L{1} = 'CLASS A';
LH(2) = plot(nan, nan, 'or');
L{2} = 'CLASS B';
LH(3) = plot(nan, nan, 'og');
L{3} = 'CLASS C';
legend(LH, L, 'FontSize',12, 'Location', 'northeast');
xlabel('w1 values','FontSize',12, 'Fontweight', 'bold');
yticks([1 2 3])
yticklabels({'Class A','Class B','Class C'})
hold off

%% Plot the test and training data in 2D


figure(2)
subplot(1,2,1)
plot(proj_w1(1,:),proj_w2(1,:), 'bo')
hold on;
plot(class_1(1),class_1(2), 'o','MarkerFaceColor', 'b')
hold on;
plot(proj_w1(2,:),proj_w2(2,:), 'ro')
```

```
hold on;
plot(class_2(1),class_2(2), 'o','MarkerFaceColor', 'r')
hold on;
plot(proj_w1(3,:),proj_w2(3,:), 'go')
hold on;
plot(class_3(1),class_3(2), 'o','MarkerFaceColor', 'g')
hold on;
title('Test 3: training data', 'Fontsize', 16)
xlabel('w1 values','FontSize',12, 'Fontweight', 'bold');
ylabel('w2 values','FontSize',12, 'Fontweight', 'bold');
LH(1) = plot(nan, nan, 'ob');
L{1} = 'CLASS A';
LH(2) = plot(nan, nan, 'or');
L{2} = 'CLASS B';
LH(3) = plot(nan, nan, 'og');
L{3} = 'CLASS C';
legend(LH, L, 'FontSize',12, 'Location', 'southeast');
hold off

% Plot test data 2D

subplot(1,2,2)
plot(means(1,1),means(1,2), 'o','MarkerFaceColor', 'b')
hold on;
plot(means(2,1),means(2,2), 'o','MarkerFaceColor', 'r')
hold on;
plot(means(3,1),means(3,2), 'o','MarkerFaceColor', 'g')
hold on;
plot(pval_a(1,:),pval_a(2,:), 'bo')
hold on;
plot(pval_b(1,:),pval_b(2,:), 'ro')
hold on;
plot(pval_c(1,:),pval_c(2,:), 'go')
title('Test 3: test data', 'Fontsize', 16)
xlabel('w1 values','FontSize',12, 'Fontweight', 'bold');
ylabel('w2 values','FontSize',12, 'Fontweight', 'bold');
LH(1) = plot(nan, nan, 'ob');
L{1} = 'CLASS A';
LH(2) = plot(nan, nan, 'or');
L{2} = 'CLASS B';
LH(3) = plot(nan, nan, 'og');
L{3} = 'CLASS C';
legend(LH, L, 'FontSize',12, 'Location','southeast');
hold off


%% Plot histograms of test data 1D

% bins = 1;
%
% % Plot histogram of test data with thresholds
% figure(3)
% subplot(1,3,1)
% histogram(pval_a(1,:),bins);
```

```
% xlim([-500 1500])
% hold on,
% plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
% hold on
% plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
% xlabel('class 2 range              class 1 range              class 3 range',...
%     'Fontsize', 10, 'Fontweight', 'bold')
% title('Test 1: class 1 data', 'Fontsize', 14)
% hold off
%
% subplot(1,3,2)
% histogram(pval_b(1,:),bins);
% xlim([-500 1500])
% hold on,
% plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
% hold on
% plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
% xlabel('class 2 range              class 1 range              class 3 range',...
%     'Fontsize', 10, 'Fontweight', 'bold')
% title('Test 1: class 2 data', 'Fontsize', 14)
% hold off
%
% subplot(1,3,3)
% histogram(pval_c(1,:),bins);
% xlim([-500 1500])
% hold on,
% plot([thresholds(1,1);thresholds(1,1)],get(gca, 'ylim'),'r')
% hold on
% plot([thresholds(1,2);thresholds(1,2)],get(gca, 'ylim'),'r')
% xlabel('class 2 range              class 1 range              class 3 range',...
%     'Fontsize', 10, 'Fontweight', 'bold')
% title('Test 1: class 3 data', 'Fontsize', 14)
% hold off

%% Classify using 1D

% Classify using first projection vector

function classes = classify_1D(pval,threshold,rank)

    dim = size(pval);
    n = dim(2);
    classes = zeros(1,n);

    for j=1:n
        if pval(j)<threshold(2) && pval(j)>threshold(1)
            classes(j) = rank(2);
        elseif pval(j)>threshold(2)
            classes(j) = rank(3);
        else
            classes(j) = rank(1);
```

```
            end
        end
end


%% Classify 2D

% Classify using both projection vectors

function classes = classify_2D(pvals,means)

        dim = size(pvals);
        n = dim(2);
        classes = zeros(1,n);

        % for every spectrogram in the data
        for j=1:n
            A = [pvals(:,j)'; means(1,:)];
            B = [pvals(:,j)'; means(2,:)];
            C = [pvals(:,j)'; means(3,:)];

            % compute the minimumm distance to a class mean
            distances = [pdist(A,'euclidean') ...
            pdist(B,'euclidean')...
            pdist(C,'euclidean')];

            [~, min_idx] = min(distances);
            classes(j) = min_idx;
        end

end


function [U,S,V,thresholds,d_arr,w_arr,rank,proj_w1,proj_w2] = trainer(s1,s2,s3,feature)

        % a 800k by 30 matrix of the spectrograms from all 3 groups
        X = [s1 s2 s3];

        ns1 = size(s1,2);
        ns2 = size(s2,2);
        ns3 = size(s3,2);

        [U,S,V] = svd(X,'econ');

        songs = S*V'; % projection onto principal components

        % choose how many singular values we want
        U = U(:,1:feature);

        % separate the songs now that they are
        % in the principal component basis
        song_1 = songs(1:feature,1:ns1);
        song_2 = songs(1:feature,ns1+1:ns1+ns2);
```

```
    song_3 = songs(1:feature,ns1+ns2+1:ns1+ns2+ns3);

    ms_all = mean(songs(1:feature,:),2);
    ms1 = mean(song_1,2);
    ms2 = mean(song_2,2);
    ms3 = mean(song_3,2);


    Sw = 0; % within class variances
    for k=1:ns1
        Sw = Sw + (song_1(:,k)-ms1)*(song_1(:,k)-ms1)';
    end
    for k=1:ns2
        Sw = Sw + (song_2(:,k)-ms2)*(song_2(:,k)-ms2)';
    end
    for k=1:ns3
        Sw = Sw + (song_3(:,k)-ms3)*(song_3(:,k)-ms3)';
    end

    Sb = (ms1-ms_all)*(ms1-ms_all)'; % between class
    Sb = Sb + (ms2-ms_all)*(ms2-ms_all)';
    Sb = Sb + (ms3-ms_all)*(ms3-ms_all)';


    % get the eigenvalue
    [V2,D] = eig(Sb,Sw); % linear discriminant analysis
    max(max(D));

    [d1,ind] = max(abs(diag(D)));

    % eigenvector with max eigenvalue
    w1 = V2(:,ind);
    w1 = w1/norm(w1,2);

    D(:,ind) = [];
    [d2,ind] = max(abs(diag(D)));

    d_arr = [d1,d2];

    % eigenvector with 2nd max eigenvalue
    w2 = V2(:,ind);
    w2 = w2/norm(w2,2);

    w_arr = [w1 w2]';

    % project the song data onto w1
    sort_s1a = w1'*song_1;
    sort_s2a = w1'*song_2;
    sort_s3a = w1'*song_3;

    proj_w1 = [sort_s1a; sort_s2a; sort_s3a];

    % project the song data onto w2
    sort_s1b = w2'*song_1;
```

```
    sort_s2b = w2'*song_2;
    sort_s3b = w2'*song_3;

    proj_w2 = [sort_s1b; sort_s2b; sort_s3b];


    % figure out which means are 'neighbors'

    mv_a = [mean(sort_s1a) mean(sort_s2a) mean(sort_s3a)];

    for j=1:3
        if mv_a(j) == max(mv_a)
            % max_val = mv_a(j);
            max_class_a = j;
        elseif mv_a(j) == min(mv_a)
            % min_val = mv_a(j);
            min_class_a = j;
        else
            % med_val = mv_a(j);
            med_class_a = j;
        end
    end

    rank = [min_class_a med_class_a max_class_a];

    vsm = [sort_s1a; sort_s2a; sort_s3a];

    % sort all values from lowest to highest
    sort_max_a = sort(vsm(max_class_a,:));
    sort_med_a = sort(vsm(med_class_a,:));
    sort_min_a = sort(vsm(min_class_a,:));

%     thresholds(1,1) = (mean(sort_med_a ) - mean(sort_min_a))/2;
%     thresholds(1,2) = (mean(sort_max_a ) - mean(sort_med_a))/2;
    t1a = length(sort_max_a);
    t2a = 1;
    while (t1a > 0 && sort_max_a(t1a)>sort_med_a(t2a))
    t1a = t1a-1;
    t2a = t2a+1;
    end
    if t1a == 0
    thresholds(1,2) = (sort_max_a(1)+sort_med_a(length(sort_max_a)
        ))/2;
    else
    thresholds(1,2) = (sort_max_a(t1a)+sort_med_a(t2a))/2;
    end


    t1b = length(sort_med_a);
    t2b = 1;
    while (t1b > 0 && sort_med_a(t1b)>sort_min_a(t2b))
    t1b = t1b-1;
    t2b = t2b+1;
    end
```

```

    if t1b == 0
    thresholds(1,1) = (sort_med_a(1)+sort_min_a(length(sort_med_a)))/2;
    else
    thresholds(1,1) = (sort_med_a(t1b)+sort_min_a(t2b))/2;
    end

    % figure out which means are 'neighbors'

    mv_b = [mean(sort_s1b) mean(sort_s2b) mean(sort_s3b)];

    for j=1:3
        if mv_b(j) == max(mv_b)
            % max_val = mv_a(j);
            max_class_b = j;
        elseif mv_b(j) == min(mv_b)
            % min_val = mv_a(j);
            min_class_b = j;
        else
            % med_val = mv_a(j);
            med_class_b = j;
        end
    end

    vsm = [sort_s1b; sort_s2b; sort_s3b];

    % sort all values from lowest to highest
    sort_max_b = sort(vsm(max_class_b,:));
    sort_med_b = sort(vsm(med_class_b,:));
    sort_min_b = sort(vsm(min_class_b,:));



%     thresholds(2,1) = (mean(sort_med_b ) - mean(sort_min_b))/2;
%     thresholds(2,2) = (mean(sort_max_b ) - mean(sort_med_b))/2;


    t1a = length(sort_max_b);
    t2a = 1;
    while (t1a > 0 && sort_max_b(t1a)>sort_med_b(t2a))
    t1a = t1a-1;
    t2a = t2a+1;
    end
    if t1a == 0
    thresholds(2,2) = (sort_max_b(1)+sort_med_b(length(sort_max_b)))/2;
    else
    thresholds(2,2) = (sort_max_b(t1a)+sort_med_b(t2a))/2;
    end


    t1b = length(sort_med_b);
    t2b = 1;
```

```
    while (t1b > 0 && sort_med_b(t1b)>sort_min_b(t2b))
    t1b = t1b-1;
    t2b = t2b+1;
    end

    if t1b == 0
    thresholds(2,1) = (sort_med_b(1)+sort_min_b(length(sort_med_b)))/2;
    else
    thresholds(2,1) = (sort_med_b(t1b)+sort_min_b(t2b))/2;
    end

end
```