

# Homework 4: Music Classification

Yilin Li

March 6 2020

## Abstract

This project applies the Singular Value Decomposition (SVD) and develops my own Linear Discrimination Analysis (LDA) algorithm to do machine learning. In this project, I use my own train function to classify the music and then calculate the accuracy in correctly identifying 5-second sound clips of songs.

## Sec.I. Introduction and Overview

Nowadays, machine learning is a really useful tool in many fields. The computer can be trained to learn from training data and make decision and predictions on the testing data. In this project, I develops my own Linear Discrimination Analysis (LDA) algorithm to train the computer to classify a given piece of music by sampling a 5 second clip. There are three kinds of tests in this project. The first test classifies three different bands of different genres. The second test classifies three different bands of same genre. The third test classifies three genres by choosing various bands within each genre. At the end, I calculate the accuracy in correctly identifying 5-second sound clips of songs.

The Theoretical Background section introduces the theorems used in the project.

The Algorithm Implementation and Development section lists the algorithms that were implemented in this project and how they were developed.

The Computational Results section shows the computational results of MATLAB and analyzes it.

The Summary and Conclusion section summarizes the project and makes a conclusion.

The MATLAB functions and implementations used in this project were attached in Appendix A. The MATLAB codes were attached in Appendix B.

## Sec.II. Theoretical Background

## The Singular Value Decomposition (SVD)

Let the transformation of a vector  $x$  be interpreted as a matrix  $A$  geometrically. The transformation can always be decomposed in various orthogonal directions. The Singular Value Decomposition reforms the matrix  $A$  into:

$$A = U\Sigma V^* \quad (1)$$

If  $A$  is an  $n \times m$  matrix, Then the columns of  $U \in \mathbb{C}^{m \times m}$  are the left singular vectors of  $A$ , the rows of  $V \in \mathbb{C}^{n \times n}$  are the right singular vectors of  $A$ , and the diagonal elements of  $\Sigma \in \mathbb{R}^{n \times m}$  are the singular values of  $A$ . The singular values are always non-negative and ordered from largest to smallest. The number of nonzero singular values is the rank (dimension of the range of  $A$ ) of  $A$ . Let  $r = \text{rank}(A)$ . Then,  $U$  is a basis for the range of  $A$ , and  $V$  is a basis for the null space of  $A$ . Singular values are the absolute values of the eigenvalues, and singular vectors equals to the eigenvectors.

## The Linear Discrimination Analysis (LDA)

The goal of LDA is two-fold: find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. For a two-class LDA, the above idea results in consideration of the following mathematical formulation.

$$w = \underset{w}{\operatorname{argmax}} \frac{w^T S_B w}{w^T S_W w} \quad (2)$$

where  $w$  is the projection,  $S_B$  is the scatter matrices for between-class, and  $S_W$  is the scatter matrices for within-class:

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \quad (3)$$

$$S_W = \sum_{j=1}^2 \sum_x (x - \mu_j)(x - \mu_j)^T \quad (4)$$

These quantities measure the variance of the data sets and the variance of the different in the means. The maximum eigenvalues  $\lambda$  and its associated eigenvector gives the quantity of interest and the projection basis.

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

However, since we have three classes for every test, I use multi-class LDA, so, the scatter matrices for between-class changes to:

$$S_B = \frac{1}{C} \sum_{i=1}^C (\mu_i - \mu)(\mu_i - \mu)^T \quad (6)$$

where  $C$  is the number of classes, which is 3 in this project.

## Sec.III. Algorithm Implementation and Development

- Find music on *Free Music Archive* and store the url(s).
- Create an empty matrix **A** to store the spectrogram of each music.
- In order to create the training data, create a **for** loop to do the following steps on every song.
  - Use **webread** to load the song directly into MATLAB from the website.
  - Re-sample the data by taking every other point in order to keep the data sizes more manageable.
  - Since the music files are loaded as *stereo* which has two *channels*, I average the two in order to get a 1D audio signal (convert from stereo to mono).
  - Remove leading and trailing 0s
  - Use a **for** loop to take 25 pieces of data per song, so I have 50 pieces of data for each band. For each piece of data, take the **absolute** value of the **spectrogram** of the data, and **reshape** the data into a column vector and store it into the matrix **A** that I created before.
- Repeat the same **for** loop to create the testing data.
- Separate the training data and testing data for different classes.
- Create the LDA algorithm which is represented as a **function** called **music\_trainer**, but the function showed at the end of the MATLAB code because the **function** must be at the end of the script.
  - Get the size of each training data.
  - Gather all the training data into a big matrix A.
  - Apply SVD to produce an economy-size decomposition of matrix A (get rid of all zero values), and I do SVD on spectrogram because the spectrogram is in frequency space and that is where the different genre of music differs from each other.
  - Since only a limited number of the principal components are considered in the feature detection (I take 20 components), I only keep a limited number of columns of U and rows of  $\Sigma V$  are extracted (projection onto principal components).
  - New variables **typr1**, **typr2**, **typr3** are generated giving the strength of their projections.
  - Calculate the mean value of each class and the mean of the mean value for each class.
  - Apply the  $S_W$  function and the  $S_B$  function.
  - Do LDA and apply the  $w$  function.

- Get the new vector for each classes by multiplying  $w'$  with the original vector.
  - Calculate the mean value of each new vector and sort the element of every new vector.
  - Print out the mean value to figure the min, middle, and max mean value, and then build the statistical threshold based on them.
- Apply the **music\_trainer** function to the three test data.
  - Apply the PCA projection.
  - Apply the LDA projection.
  - Use the **threshold1** and **threshold2** to distinguish the band (genre) and store the result as 0, 1, 2.
  - Create **hiddenlabels** to represent the true band (genre) by using the 0, 1, 2 respectively.
  - Use a **for** loop to check the results and calculate the rate of success. To be specific, set the successful classification to be 0, add 1 when a single piece is classified successfully (the number of result is same as the true value), and then divided by the total number of the pieces to get the accuracy.

## Sec.IV. Computational Results

### Test 1: Band Classification

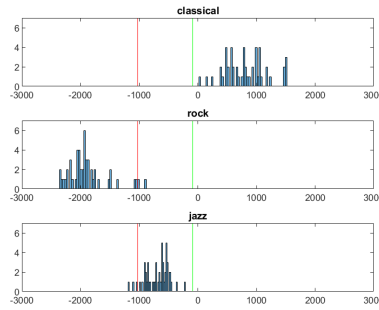


Figure 1: Histogram of the statistics associated with the classical, rock, and jazz data once projected onto the LDA basis

This red line is the numerically computed decision threshold 1. The green line is the numerically computed decision threshold 2. Thus a new music clip would be projected onto the LDA basis, and a decision would be made concerning the music clip depending upon which side of the threshold line it sits. Test 2 and test 3 are same as this.

From the result, it is shown that the accuracy for my LDA algorithm on test 1 is 50.67%, which means that about 38 out of 75 test data were distinguished successfully.

## Test 2: The Case for Same Genre

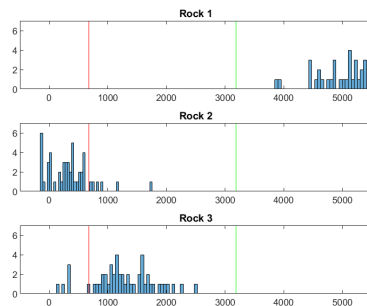


Figure 2: Histogram of the statistics associated with three rock bands data once projected onto the LDA basis

From the result, it is shown that the accuracy for my LDA algorithm on test 2 is 64%, which means that about 48 out of 75 test data were distinguished successfully. This result was unexpected because distinguishing different bands from the same genre should make the testing and separation much more challenging. Therefore, the accuracy should be lower than test 1. However, I found that the three rock bands that I chose for test 2 were really different from each other even they were from the same genre. Since rock has many different types, this makes sense.

## Test 3: Genre Classification

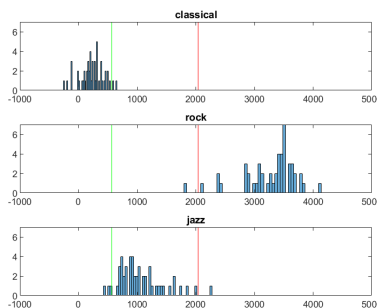


Figure 3: Histogram of the statistics associated with the classical, rock, and jazz data once projected onto the LDA basis

From the result, it is shown that the accuracy for my LDA algorithm on test 3 is 48%, which means that about 36 out of 75 test data were distinguished successfully. This result was expected because when I use various bands within each genre as the training data, different bands might make the testing and separation harder. For instance, the two rock bands I found for this test were different from each other. Compare to more training data for the same band within different genre(test 1), it is harder to distinguish the genre of a band by using different bands as training data.

## Sec.V. Summary and Conclusions

For this project, I developed classification models for music identification by developing my own LDA algorithm. I performed three different types of tests. In each test, I applied SVD analysis of the spectrogram of the music I picked. I created training data and testing data, and then applied my LDA model on them. I found that my model performed better if the genres or the bands within the same genre were very different, and worse when they were more similar within the different genre or more different within the same genre (for test 3). This result is expected. The accuracy of classification highly depends on the nature of the music. Additionally, increasing the number of the training data might also increase the accuracy depends on the similarity and the difference between the music.

## Appendix A

- **[data,Fs] = webread(url)** reads audio data from the web service specified by url and returns the audio data in data.
- **L = length(X)** returns the length of the largest array dimension in X.
- **k = find(X)** returns a vector containing the linear indices of each nonzero element in array X.
- **abs(X)** returns the absolute value of each element in array X.
- **s = spectrogram(x)** returns the short-time Fourier transform of the input signal, x. Each column of s contains an estimate of the short-term, time-localized frequency content of x.
- **B = reshape(A,sz)** reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix.
- **X = zeros(sz1,...,szN)** returns an sz1-by-...-by-szN array of zeros where sz1,...,szN indicate the size of each dimension.
- **X = ones(sz1,...,szN)** returns an sz1-by-...-by-szN array of ones where sz1,...,szN indicate the size of each dimension.
- **function [y1,...,yN] = myfun(x1,...,xM)** declares a function named myfun that accepts inputs x1,...,xM and returns outputs y1,...,yN.
- **sz = size(A)** returns a row vector whose elements are the lengths of the corresponding dimensions of A.
- **[U,S,V] = svd(A)** performs a singular value decomposition of matrix A, such that  $A = U*S*V'$ .
- **M = mean(A)** returns the mean of the elements of A along the first array dimension whose size does not equal 1.

- **[V,D] = eig(A,B)** returns diagonal matrix D of generalized eigenvalues and full matrix V whose columns are the corresponding right eigenvectors, so that  $A*V = B*V*D$ .
- **M = max(A)** returns the maximum elements of an array.
- **D = diag(v)** returns a square diagonal matrix with the elements of vector v on the main diagonal.
- **n = norm(v,p)** returns the generalized vector p-norm.
- **B = sort(A)** sorts the elements of A in ascending order.
- **histogram(X,nbins)** uses a number of bins specified by the scalar, nbins.
- **plot(X,Y)** creates a 2-D line plot of the data in Y versus the corresponding values in X.

## Appendix B. MATLAB codes

### Test 1

```

1 clear; close all; clc;
2
3 % Homework 4
4 % Test 1: Band Classification
5
6 str = ["https://files.freemusicarchive.org/storage-
       freemusicarchive-org/music/ccCommunity/Yakov_Golman/
       Piano__orchestra_1/Yakov_Golman_-_07_-_Rainbow.mp3", "https://
       files.freemusicarchive.org/storage-freemusicarchive-org/music/
       ccCommunity/Yakov_Golman/Piano__orchestra_1/Yakov_Golman_-_06_-_
       _Prelude.mp3", "https://files.freemusicarchive.org/storage-
       freemusicarchive-org/music/none_given/Scott_Holmes/
       Scott_Holmes_-_Singles/Scott_Holmes_-_Driven_To_Success.mp3", "
       https://files.freemusicarchive.org/storage-freemusicarchive-org
       /music/none_given/Scott_Holmes/Scott_Holmes_-_Singles/
       Scott_Holmes_-_Aspire.mp3", "https://files.freemusicarchive.org/
       storage-freemusicarchive-org/music/Ziklibrenbib/Mela/Mela_two/
       Mela_-_03_-_Horrible.mp3", "https://files.freemusicarchive.org/
       storage-freemusicarchive-org/music/Ziklibrenbib/Mela/Mela_two/
       Mela_-_02_-_Free_Time.mp3" ];
7
8
9 A = [];
10

```

```

11 for jj = 1:length(str)
12
13     [y,Fs] = webread(str(jj));
14     Fs = Fs/2;
15     y = y(1:2:end,:);
16     song = [];
17     for j = 1:length(y)
18         song(j,1) = (y(j,1) + y(j,2))/2;
19     end
20     song = song(find(song,1,'first'):find(song,1,'last'));
21
22
23     for k = 1:5:125
24         test = song(Fs*k : Fs*(k+5),1);
25         vector = abs(spectrogram(test));
26         vector = reshape(vector,[length(vector)*8,1]);
27         A = [A vector];
28     end
29
30 end
31
32 train1 = A(:, 1:50);
33 train2 = A(:, 51:100);
34 train3 = A(:, 101:150);
35
36 [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] =
    music_trainer(train1,train2,train3);
37
38 str2 = ["https://files.freemusicarchive.org/storage-
    freemusicarchive-org/music/ccCommunity/Yakov_Golman/
    Piano__orchestra_1/Yakov_Golman__08__Valse_orchestral.mp3","
    https://files.freemusicarchive.org/storage-freemusicarchive-org
    /music/none_given/Scott_Holmes/Scott_Holmes__Singles/
    Scott_Holmes___Teamwork.mp3","https://files.freemusicarchive.
    org/storage-freemusicarchive-org/music/Ziklibrenbib/Mela/
    Mela_two/Mela__07__The_Darker_Side.mp3"];
39
40 B = [];
41
42 for zz = 1:length(str2)
43
44     [y2,Fs2] = webread(str2(zz));
45     Fs2 = Fs2/2;
46     y2 = y2(1:2:end,:);
47     song2 = [];

```



```

48     for j = 1:length(y2)
49         song2(j,1) = (y2(j,1) + y2(j,2))/2;
50     end
51     song2 = song2(find(song2,1,'first'):find(song2,1,'last'));
52
53
54     for k = 1:5:125
55         test2 = song2(Fs2*k : Fs2*(k+5),1);
56         vector2 = abs(spectrogram(test2));
57         vector2 = reshape(vector2,[length(vector2)*8,1]);
58         B = [B vector2];
59     end
60
61 end
62
63 test1 = B(:,1:25);
64 test2 = B(:,26:50);
65 test3 = B(:,51:75);
66
67 Test = [test1 test2 test3];
68 TestMat = U' * Test; % PCA projection
69 pval = w' * TestMat; % LDA projection
70
71 % Rock = 0, Jazz = 1, Classical = 2
72 ResVec = [];
73 for kk = 1:length(pval)
74     if pval(kk) <= threshold1
75         ResVec(kk) = 0;
76     elseif threshold1 < pval(kk) <= threshold2
77         ResVec(kk) = 1;
78     else
79         ResVec(kk) = 2;
80     end
81 end
82 ResVec = ResVec';
83 hiddenlabels = [2*ones(25,1);zeros(25,1);ones(25,1)];
84 disp('Rate of success');
85 sucRate = 0;
86 for jj = 1:75
87     if ResVec(jj) == hiddenlabels(jj)
88         sucRate = sucRate + 1;
89     end
90 end
91
92 sucRate = sucRate / 75

```

```

93
94 function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,
    sorttype3] = music_trainer(train1_0,train2_0,train3_0)
95 n1 = size(train1_0,2);
96 n2 = size(train2_0,2);
97 n3 = size(train3_0,2);
98 A = [train1_0 train2_0 train3_0];
99 [U,S,V] = svd(A, 'econ');
100 songs = S*V';
101 U = U(:,1:20);
102 type1 = songs(1:20, 1:n1);
103 type2 = songs(1:20, (n1+1):(n1+n2));
104 type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));
105
106 m1 = mean(type1, 2);
107 m2 = mean(type2, 2);
108 m3 = mean(type3, 2);
109 m = (m1+m2+m3)/3;
110
111 Sw = 0; % within class variances
112 for k=1:n1
113     Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';
114 end
115 for k=1:n2
116     Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
117 end
118 for k=1:n3
119     Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
120 end
121
122 Sb = ((m1-m)*(m1-m)' + (m2-m)*(m2-m)' + (m3-m)*(m3-m)')/3; % between
    class
123
124 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
125 [~,ind] = max(abs(diag(D)));
126 w = V2(:,ind); w = w/norm(w,2);
127
128 vtype1 = w'*type1;
129 vtype2 = w'*type2;
130 vtype3 = w'*type3;
131
132 mean(vtype1); % max
133 mean(vtype2); % min
134 mean(vtype3); % middle
135

```

```

136 sorttype1 = sort(vtype1);
137 sorttype2 = sort(vtype2);
138 sorttype3 = sort(vtype3);
139
140 t1 = length(sorttype2);
141 t2 = 1;
142 while sorttype2(t1)>sorttype3(t2)
143     t1 = t1 - 1;
144     t2 = t2 + 1;
145 end
146 threshold1 = (sorttype2(t1)+sorttype3(t2))/2;
147
148 t3 = length(sorttype3);
149 t4 = 1;
150 while sorttype3(t3)>sorttype1(t4)
151     t3 = t3 - 1;
152     t4 = t4 + 1;
153 end
154 threshold2 = (sorttype3(t3)+sorttype1(t4))/2;
155
156 subplot(3,1,1)
157 histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
158 hold on, plot([threshold2 threshold2], [0 7], 'g');
159 set(gca, 'Xlim', [-3000 3000], 'Ylim', [0 7])
160 title('classical')
161 subplot(3,1,2)
162 histogram(sorttype2,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
163 hold on, plot([threshold2 threshold2], [0 7], 'g');
164 set(gca, 'Xlim', [-3000 3000], 'Ylim', [0 7])
165 title('rock')
166 subplot(3,1,3)
167 histogram(sorttype3,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
168 hold on, plot([threshold2 threshold2], [0 7], 'g');
169 set(gca, 'Xlim', [-3000 3000], 'Ylim', [0 7])
170 title('jazz')
171
172 end

```

## Test 2

```

1 clear; close all; clc;
2

```

```

3 % Homework 4
4 % Test 2: The Case for Same Genre
5
6 str = ["https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Ryan_Andersen/
Never_Sleep_-_Indie_Rock/Ryan_Andersen_-_Shmail_Nail.mp3", "
https://files.freemusicarchive.org/storage-freemusicarchive-org
/music/ccCommunity/Ryan_Andersen/Never_Sleep_-_Indie_Rock/
Ryan_Andersen_-_London_Calling.mp3", "https://files.
freemusicarchive.org/storage-freemusicarchive-org/music/
none_given/Scott_Holmes/Scott_Holmes_-_Singles/Scott_Holmes_-_
_Driven_To_Success.mp3", "https://files.freemusicarchive.org/
storage-freemusicarchive-org/music/none_given/Scott_Holmes/
Scott_Holmes_-_Singles/Scott_Holmes_-_Aspire.mp3", "https://
files.freemusicarchive.org/storage-freemusicarchive-org/music/
Ziklibrenbib/The_Inventors/Counting_backwards/The_Inventors_-_
_06_-_Fire.mp3", "https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/Ziklibrenbib/The_Inventors/
Counting_backwards/The_Inventors_-_04_-_Melon.mp3"];
7
8 A = [];
9
10 for jj = 1:length(str)
11
12     [y,Fs] = webread(str(jj));
13     Fs = Fs/2;
14     y = y(1:2:end,:);
15     song = [];
16     for j = 1:length(y)
17         song(j,1) = (y(j,1) + y(j,2))/2;
18     end
19     song = song(find(song,1,'first'):find(song,1,'last'));
20
21
22     for k = 1:5:125
23         test = song(Fs*k : Fs*(k+5),1);
24         vector = abs(spectrogram(test));
25         vector = reshape(vector,[length(vector)*8,1]);
26         A = [A vector];
27     end
28
29 end
30
31 train1 = A(:, 1:50);
32 train2 = A(:, 51:100);

```

```

33 train3 = A(:, 101:150);
34
35 [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] =
    music_trainer(train1,train2,train3);
36
37 str2 = ["https://files.freemusicarchive.org/storage-
    freemusicarchive-org/music/ccCommunity/Ryan_Andersen/
    Never_Sleep_-_Indie_Rock/Ryan_Andersen_-_Cosmic_Sleep.mp3", "
    https://files.freemusicarchive.org/storage-freemusicarchive-org
    /music/none_given/Scott_Holmes/Scott_Holmes_-_Singles/
    Scott_Holmes_-_Teamwork.mp3", "https://files.freemusicarchive.
    org/storage-freemusicarchive-org/music/Ziklibrenbib/
    The_Inventors/Counting_backwards/The_Inventors_-_03_-_
    _Still_Sailor.mp3"];
38
39 B = [];
40
41 for zz = 1:length(str2)
42
43     [y2,Fs2] = webread(str2(zz));
44     Fs2 = Fs2/2;
45     y2 = y2(1:2:end,:);
46     song2 = [];
47     for j = 1:length(y2)
48         song2(j,1) = (y2(j,1) + y2(j,2))/2;
49     end
50     song2 = song2(find(song2,1,'first'):find(song2,1,'last'));
51
52
53     for k = 1:5:125
54         test2 = song2(Fs2*k : Fs2*(k+5),1);
55         vector2 = abs(spectrogram(test2));
56         vector2 = reshape(vector2,[length(vector2)*8,1]);
57         B = [B vector2];
58     end
59
60 end
61
62 test1 = B(:,1:25);
63 test2 = B(:,26:50);
64 test3 = B(:,51:75);
65
66 Test = [test1 test2 test3];
67 TestMat = U' * Test; % PCA projection
68 pval = w' * TestMat; % LDA projection

```

```

69
70 % Rock = 0, Jazz = 1, Classical = 2
71 ResVec = [];
72 for kk = 1:length(pval)
73     if pval(kk) <= threshold1
74         ResVec(kk) = 0;
75     elseif threshold1 < pval(kk) <= threshold2
76         ResVec(kk) = 1;
77     else
78         ResVec(kk) = 2;
79     end
80 end
81 ResVec = ResVec';
82 hiddenlabels = [2*ones(25,1); zeros(25,1); ones(25,1)];
83 disp('Rate of success');
84 sucRate = 0;
85 for jj = 1:75
86     if ResVec(jj) == hiddenlabels(jj)
87         sucRate = sucRate + 1;
88     end
89 end
90
91 sucRate = sucRate / 75
92
93 function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,
94     sorttype3] = music_trainer(train1_0,train2_0,train3_0)
95 n1 = size(train1_0,2);
96 n2 = size(train2_0,2);
97 n3 = size(train3_0,2);
98 A = [train1_0 train2_0 train3_0];
99 [U,S,V] = svd(A,'econ');
100 songs = S*V';
101 U = U(:,1:20);
102 type1 = songs(1:20, 1:n1);
103 type2 = songs(1:20, (n1+1):(n1+n2));
104 type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));
105 m1 = mean(type1, 2);
106 m2 = mean(type2, 2);
107 m3 = mean(type3, 2);
108 m = (m1+m2+m3)/3;
109
110 Sw = 0; % within class variances
111 for k=1:n1
112     Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';

```

```

113 end
114 for k=1:n2
115     Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
116 end
117 for k=1:n3
118     Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
119 end
120
121 Sb = ((m1-m)*(m1-m)'+(m2-m)*(m2-m)'+(m3-m)*(m3-m)')/3; % between
    class
122
123 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
124 [~,ind] = max(abs(diag(D)));
125 w = V2(:,ind); w = w/norm(w,2);
126
127 vtype1 = w'*type1;
128 vtype2 = w'*type2;
129 vtype3 = w'*type3;
130
131 mean(vtype1) % max
132 mean(vtype2) % min
133 mean(vtype3) % middle
134
135 sorttype1 = sort(vtype1);
136 sorttype2 = sort(vtype2);
137 sorttype3 = sort(vtype3);
138
139 t1 = length(sorttype2);
140 t2 = 1;
141 while sorttype2(t1)>sorttype3(t2)
142     t1 = t1 - 1;
143     t2 = t2 + 1;
144 end
145 threshold1 = (sorttype2(t1)+sorttype3(t2))/2;
146
147 t3 = length(sorttype3);
148 t4 = 1;
149 while sorttype3(t3)>sorttype1(t4)
150     t3 = t3 - 1;
151     t4 = t4 + 1;
152 end
153 threshold2 = (sorttype3(t3)+sorttype1(t4))/2;
154
155 subplot(3,1,1)
156 histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0

```

```

    7], 'r')
157 hold on, plot([threshold2 threshold2], [0 7], 'g');
158 set(gca, 'Xlim', [-500 5500], 'Ylim', [0 7])
159 title('Rock 1')
160 subplot(3,1,2)
161 histogram(sorttype2,50); hold on, plot([threshold1 threshold1], [0
    7], 'r')
162 hold on, plot([threshold2 threshold2], [0 7], 'g');
163 set(gca, 'Xlim', [-500 5500], 'Ylim', [0 7])
164 title('Rock 2')
165 subplot(3,1,3)
166 histogram(sorttype3,50); hold on, plot([threshold1 threshold1], [0
    7], 'r')
167 hold on, plot([threshold2 threshold2], [0 7], 'g');
168 set(gca, 'Xlim', [-500 5500], 'Ylim', [0 7])
169 title('Rock 3')
170
171 end

```

### Test 3

```

1 clear; close all; clc;
2
3 % Homework 4
4 % Test 3: Genre Classification
5 str = ["https://files.freemusicarchive.org/storage-
    freemusicarchive-org/music/ccCommunity/Yakov_Golman/
    Piano__orchestra_1/Yakov_Golman_-_07_-_Rainbow.mp3", "https://
    files.freemusicarchive.org/storage-freemusicarchive-org/music/
    Music_for_Video/Lloyd_Rodgers/
    Cartesian_Reunion_Memorial_Orchestra_the_little_prince-
    a_ballet_in_two_acts/Lloyd_Rodgers_-_08_-_
    _On_Questions_of_Responsibility_Act_II.mp3", "https://files.
    freemusicarchive.org/storage-freemusicarchive-org/music/
    none_given/Scott_Holmes/Scott_Holmes_-_Singles/Scott_Holmes_-_
    _Driven_To_Success.mp3", "https://files.freemusicarchive.org/
    storage-freemusicarchive-org/music/Ziklibrenbib/The_Inventors/
    Counting_backwards/The_Inventors_-_04_-_Melon.mp3", "https://
    files.freemusicarchive.org/storage-freemusicarchive-org/music/
    Ziklibrenbib/Mela/Mela_two/Mela_-_03_-_Horrible.mp3", "https://
    files.freemusicarchive.org/storage-freemusicarchive-org/music/
    ccCommunity/Dee_Yan-Key/facts_of_life/Dee_Yan-Key_-_08_-_Blue.
    mp3"];
6
7 A = [];

```



```

8
9  for jj = 1:length(str)
10
11     [y,Fs] = webread(str(jj));
12     Fs = Fs/2;
13     y = y(1:2:end,:);
14     song = [];
15     for j = 1:length(y)
16         song(j,1) = (y(j,1) + y(j,2))/2;
17     end
18     song = song(find(song,1,'first'):find(song,1,'last'));
19
20
21     for k = 1:5:125
22         test = song(Fs*k : Fs*(k+5),1);
23         vector = abs(spectrogram(test));
24         vector = reshape(vector,[length(vector)*8,1]);
25         A = [A vector];
26     end
27
28 end
29
30 train1 = A(:, 1:50);
31 train2 = A(:, 51:100);
32 train3 = A(:, 101:150);
33
34 [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] =
    music_trainer(train1,train2,train3);
35
36 str2 = ["https://files.freemusicarchive.org/storage-
    freemusicarchive-org/music/ccCommunity/Yakov_Golman/
    Piano__orchestra_1/Yakov_Golman_-_08_-_Valse_orchestral.mp3","
    https://files.freemusicarchive.org/storage-freemusicarchive-org
    /music/none_given/Scott_Holmes/Scott_Holmes_-_Singles/
    Scott_Holmes_-_Teamwork.mp3","https://files.freemusicarchive.
    org/storage-freemusicarchive-org/music/Ziklibrenbib/Mela/
    Mela_two/Mela_-_07_-_The_Darker_Side.mp3"];
37
38 B = [];
39
40 for zz = 1:length(str2)
41
42     [y2,Fs2] = webread(str2(zz));
43     Fs2 = Fs2/2;
44     y2 = y2(1:2:end,:);

```

```

45     song2 = [];
46     for j = 1:length(y2)
47         song2(j,1) = (y2(j,1) + y2(j,2))/2;
48     end
49     song2 = song2(find(song2,1,'first'):find(song2,1,'last'));
50
51
52     for k = 1:5:125
53         test2 = song2(Fs2*k : Fs2*(k+5),1);
54         vector2 = abs(spectrogram(test2));
55         vector2 = reshape(vector2,[length(vector2)*8,1]);
56         B = [B vector2];
57     end
58
59 end
60
61 test1 = B(:,1:25);
62 test2 = B(:,26:50);
63 test3 = B(:,51:75);
64
65 Test = [test1 test2 test3];
66 TestMat = U' * Test; % PCA projection
67 pval = w' * TestMat; % LDA projection
68
69 % Rock = 0, Jazz = 1, Classical = 2
70 ResVec = [];
71 for kk = 1:length(pval)
72     if pval(kk) <= threshold1
73         ResVec(kk) = 0;
74     elseif threshold1 < pval(kk) <= threshold2
75         ResVec(kk) = 1;
76     else
77         ResVec(kk) = 2;
78     end
79 end
80 ResVec = ResVec';
81 hiddenlabels = [2*ones(25,1);zeros(25,1);ones(25,1)];
82 disp('Rate of success');
83 sucRate = 0;
84 for jj = 1:75
85     if ResVec(jj) == hiddenlabels(jj)
86         sucRate = sucRate + 1;
87     end
88 end
89

```

```

90 sucRate = sucRate / 75
91
92 function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,
    sorttype3] = music_trainer(train1_0,train2_0,train3_0)
93 n1 = size(train1_0,2);
94 n2 = size(train2_0,2);
95 n3 = size(train3_0,2);
96 A = [train1_0 train2_0 train3_0];
97 [U,S,V] = svd(A, 'econ');
98 songs = S*V';
99 U = U(:,1:20);
100 type1 = songs(1:20, 1:n1);
101 type2 = songs(1:20, (n1+1):(n1+n2));
102 type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));
103
104 m1 = mean(type1, 2);
105 m2 = mean(type2, 2);
106 m3 = mean(type3, 2);
107 m = (m1+m2+m3)/3;
108
109 Sw = 0; % within class variances
110 for k=1:n1
111     Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';
112 end
113 for k=1:n2
114     Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
115 end
116 for k=1:n3
117     Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
118 end
119
120 Sb = ((m1-m)*(m1-m)' + (m2-m)*(m2-m)' + (m3-m)*(m3-m)') / 3; % between
    class
121
122 [V2,D] = eig(Sb,Sw); % linear discriminant analysis
123 [~,ind] = max(abs(diag(D)));
124 w = V2(:,ind); w = w/norm(w,2);
125
126 vtype1 = w'*type1;
127 vtype2 = w'*type2;
128 vtype3 = w'*type3;
129
130 mean(vtype1) % max
131 mean(vtype2) % min
132 mean(vtype3) % middle

```

```

133
134 sorttype1 = sort(vtype1);
135 sorttype2 = sort(vtype2);
136 sorttype3 = sort(vtype3);
137
138 t1 = length(sorttype2);
139 t2 = 1;
140 while sorttype2(t1)>sorttype3(t2)
141     t1 = t1 - 1;
142     t2 = t2 + 1;
143 end
144 threshold1 = (sorttype2(t1)+sorttype3(t2))/2;
145
146 t3 = length(sorttype3);
147 t4 = 1;
148 while sorttype3(t3)>sorttype1(t4)
149     t3 = t3 - 1;
150     t4 = t4 + 1;
151 end
152 threshold2 = (sorttype3(t3)+sorttype1(t4))/2;
153
154 subplot(3,1,1)
155 histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
156 hold on, plot([threshold2 threshold2], [0 7], 'g');
157 set(gca, 'Xlim', [-1000 5000], 'Ylim', [0 7])
158 title('classical')
159 subplot(3,1,2)
160 histogram(sorttype2,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
161 hold on, plot([threshold2 threshold2], [0 7], 'g');
162 set(gca, 'Xlim', [-1000 5000], 'Ylim', [0 7])
163 title('rock')
164 subplot(3,1,3)
165 histogram(sorttype3,50);hold on, plot([threshold1 threshold1], [0
    7], 'r')
166 hold on, plot([threshold2 threshold2], [0 7], 'g');
167 set(gca, 'Xlim', [-1000 5000], 'Ylim', [0 7])
168 title('jazz')
169
170 end

```