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:

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

If A is an n*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 $r = rank(A)$. 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.

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

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

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{3}$$

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

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

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

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

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

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 ΣV 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

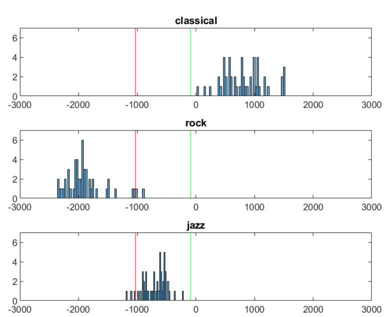

Figure 1: Histogram of the statistics associated with the classical, rock, and jazz data once projected onto the LDA basis

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

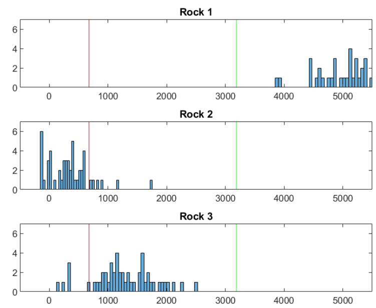

Figure 2: Histogram of the statistics associated with three rock bands data once projected onto the LDA basis

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

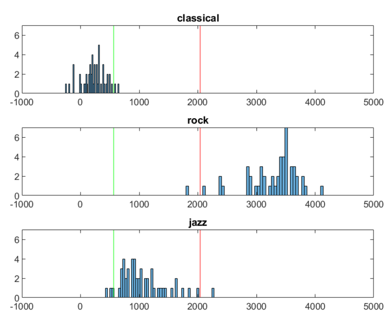

Figure 3: Histogram of the statistics associated with the classical, rock, and jazz data once projected onto the LDA basis

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

```
clear; close all; clc;

% Homework 4
% Test 1: Band Classification

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"];


A = [];

```

```
for jj = 1:length(str)

    [y,Fs] = webread(str(jj));
    Fs = Fs/2;
    y = y(1:2:end,:);
    song = [];
    for j = 1:length(y)
        song(j,1) = (y(j,1) + y(j,2))/2;
    end
    song = song(find(song,1,'first'):find(song,1,'last'));


    for k = 1:5:125
        test = song(Fs*k : Fs*(k+5),1);
        vector = abs(spectrogram(test));
        vector = reshape(vector,[length(vector)*8,1]);
        A = [A vector];
    end

end

train1 = A(:, 1:50);
train2 = A(:, 51:100);
train3 = A(:, 101:150);

[U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] = music_trainer(train1,train2,train3);

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"];

B = [];

for zz = 1:length(str2)

    [y2,Fs2] = webread(str2(zz));
    Fs2 = Fs2/2;
    y2 = y2(1:2:end,:);
    song2 = [];
```

```
    for j = 1:length(y2)
        song2(j,1) = (y2(j,1) + y2(j,2))/2;
    end
    song2 = song2(find(song2,1,'first'):find(song2,1,'last'));


    for k = 1:5:125
        test2 = song2(Fs2*k : Fs2*(k+5),1);
        vector2 = abs(spectrogram(test2));
        vector2 = reshape(vector2,[length(vector2)*8,1]);
        B = [B vector2];
    end

end

test1 = B(:,1:25);
test2 = B(:,26:50);
test3 = B(:,51:75);

Test = [test1 test2 test3];
TestMat = U' * Test;  % PCA projection
pval = w' * TestMat;  % LDA projection

% Rock = 0, Jazz = 1, Classical = 2
ResVec = [];
for kk = 1:length(pval)
    if pval(kk) <= threshold1
        ResVec(kk) = 0;
    elseif threshold1 < pval(kk) <= threshold2
        ResVec(kk) = 1;
    else
        ResVec(kk) = 2;
    end
end
ResVec = ResVec';
hiddenlabels = [2*ones(25,1);zeros(25,1);ones(25,1)];
disp('Rate of success');
sucRate = 0;
for jj = 1:75
    if ResVec(jj) == hiddenlabels(jj)
        sucRate = sucRate + 1;
    end
end

sucRate = sucRate / 75
```

```
function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] = music_trainer(train1_0,train2_0,train3_0)
n1 = size(train1_0,2);
n2 = size(train2_0,2);
n3 = size(train3_0,2);
A = [train1_0 train2_0 train3_0];
[U,S,V] = svd(A,'econ');
songs = S*V';
U = U(:,1:20);
type1 = songs(1:20, 1:n1);
type2 = songs(1:20, (n1+1):(n1+n2));
type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));

m1 = mean(type1, 2);
m2 = mean(type2, 2);
m3 = mean(type3, 2);
m = (m1+m2+m3)/3;

Sw = 0; % within class variances
for k=1:n1
    Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';
end
for k=1:n2
    Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
end
for k=1:n3
    Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
end

Sb = ((m1-m)*(m1-m)'+(m2-m)*(m2-m)'+(m3-m)*(m3-m)')/3; % between class

[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vtype1 = w'*type1;
vtype2 = w'*type2;
vtype3 = w'*type3;

mean(vtype1); % max
mean(vtype2); % min
mean(vtype3); % middle

```

```
sorttype1 = sort(vtype1);
sorttype2 = sort(vtype2);
sorttype3 = sort(vtype3);

t1 = length(sorttype2);
t2 = 1;
while sorttype2(t1)>sorttype3(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold1 = (sorttype2(t1)+sorttype3(t2))/2;

t3 = length(sorttype3);
t4 = 1;
while sorttype3(t3)>sorttype1(t4)
    t3 = t3 - 1;
    t4 = t4 + 1;
end
threshold2 = (sorttype3(t3)+sorttype1(t4))/2;

subplot(3,1,1)
histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0 7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca, 'Xlim',[-3000 3000],'Ylim',[0 7])
title('classical')
subplot(3,1,2)
histogram(sorttype2,50);hold on, plot([threshold1 threshold1], [0 7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca, 'Xlim',[-3000 3000],'Ylim',[0 7])
title('rock')
subplot(3,1,3)
histogram(sorttype3,50);hold on, plot([threshold1 threshold1], [0 7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca, 'Xlim',[-3000 3000],'Ylim',[0 7])
title('jazz')

end
```

Test 2

```
clear; close all; clc;

```

```
% Homework 4
% Test 2: The Case for Same Genre

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"];

A = [];

for jj = 1:length(str)

    [y,Fs] = webread(str(jj));
    Fs = Fs/2;
    y = y(1:2:end,:);
    song = [];
    for j = 1:length(y)
        song(j,1) = (y(j,1) + y(j,2))/2;
    end
    song = song(find(song,1,'first'):find(song,1,'last'));


    for k = 1:5:125
        test = song(Fs*k : Fs*(k+5),1);
        vector = abs(spectrogram(test));
        vector = reshape(vector,[length(vector)*8,1]);
        A = [A vector];
    end

end

train1 = A(:, 1:50);
train2 = A(:, 51:100);
```

```
train3 = A(:, 101:150);

[U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] =
    music_trainer(train1,train2,train3);

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"];

B = [];

for zz = 1:length(str2)

    [y2,Fs2] = webread(str2(zz));
    Fs2 = Fs2/2;
    y2 = y2(1:2:end,:);
    song2 = [];
    for j = 1:length(y2)
        song2(j,1) = (y2(j,1) + y2(j,2))/2;
    end
    song2 = song2(find(song2,1,'first'):find(song2,1,'last'));


    for k = 1:5:125
        test2 = song2(Fs2*k : Fs2*(k+5),1);
        vector2 = abs(spectrogram(test2));
        vector2 = reshape(vector2,[length(vector2)*8,1]);
        B = [B vector2];
    end

end

test1 = B(:,1:25);
test2 = B(:,26:50);
test3 = B(:,51:75);

Test = [test1 test2 test3];
TestMat = U' * Test;  % PCA projection
pval = w' * TestMat;  % LDA projection
```

```
% Rock = 0, Jazz = 1, Classical = 2
ResVec = [];
for kk = 1:length(pval)
    if pval(kk) <= threshold1
        ResVec(kk) = 0;
    elseif threshold1 < pval(kk) <= threshold2
        ResVec(kk) = 1;
    else
        ResVec(kk) = 2;
    end
end
ResVec = ResVec';
hiddenlabels = [2*ones(25,1);zeros(25,1);ones(25,1)];
disp('Rate of success');
sucRate = 0;
for jj = 1:75
    if ResVec(jj) == hiddenlabels(jj)
        sucRate = sucRate + 1;
    end
end

sucRate = sucRate / 75

function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] = music_trainer(train1_0,train2_0,train3_0)
n1 = size(train1_0,2);
n2 = size(train2_0,2);
n3 = size(train3_0,2);
A = [train1_0 train2_0 train3_0];
[U,S,V] = svd(A,'econ');
songs = S*V';
U = U(:,1:20);
type1 = songs(1:20, 1:n1);
type2 = songs(1:20, (n1+1):(n1+n2));
type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));

m1 = mean(type1, 2);
m2 = mean(type2, 2);
m3 = mean(type3, 2);
m = (m1+m2+m3)/3;

Sw = 0; % within class variances
for k=1:n1
    Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';
```

```
end
for k=1:n2
    Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
end
for k=1:n3
    Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
end

Sb = ((m1-m)*(m1-m)'+(m2-m)*(m2-m)'+(m3-m)*(m3-m)')/3; % between
   class

[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vtype1 = w'*type1;
vtype2 = w'*type2;
vtype3 = w'*type3;

mean(vtype1) % max
mean(vtype2) % min
mean(vtype3) % middle

sorttype1 = sort(vtype1);
sorttype2 = sort(vtype2);
sorttype3 = sort(vtype3);

t1 = length(sorttype2);
t2 = 1;
while sorttype2(t1)>sorttype3(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold1 = (sorttype2(t1)+sorttype3(t2))/2;

t3 = length(sorttype3);
t4 = 1;
while sorttype3(t3)>sorttype1(t4)
    t3 = t3 - 1;
    t4 = t4 + 1;
end
threshold2 = (sorttype3(t3)+sorttype1(t4))/2;

subplot(3,1,1)
histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0
```

```
7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-500 5500],'Ylim',[0 7])
title('Rock 1')
subplot(3,1,2)
histogram(sorttype2,50);hold on, plot([threshold1 threshold1], [0 7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-500 5500],'Ylim',[0 7])
title('Rock 2')
subplot(3,1,3)
histogram(sorttype3,50);hold on, plot([threshold1 threshold1], [0 7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-500 5500],'Ylim',[0 7])
title('Rock 3')

end
```

Test 3

```
clear; close all; clc;

% Homework 4
% Test 3: Genre Classification
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"];

A = [];
```

```
for jj = 1:length(str)

    [y,Fs] = webread(str(jj));
    Fs = Fs/2;
    y = y(1:2:end,:);
    song = [];
    for j = 1:length(y)
        song(j,1) = (y(j,1) + y(j,2))/2;
    end
    song = song(find(song,1,'first'):find(song,1,'last'));


    for k = 1:5:125
        test = song(Fs*k : Fs*(k+5),1);
        vector = abs(spectrogram(test));
        vector = reshape(vector,[length(vector)*8,1]);
        A = [A vector];
    end

end

train1 = A(:, 1:50);
train2 = A(:, 51:100);
train3 = A(:, 101:150);

[U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,sorttype3] =
    music_trainer(train1,train2,train3);

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"];

B = [];

for zz = 1:length(str2)

    [y2,Fs2] = webread(str2(zz));
    Fs2 = Fs2/2;
    y2 = y2(1:2:end,:);
```

```
    song2 = [];
    for j = 1:length(y2)
        song2(j,1) = (y2(j,1) + y2(j,2))/2;
    end
    song2 = song2(find(song2,1,'first'):find(song2,1,'last'));


    for k = 1:5:125
        test2 = song2(Fs2*k : Fs2*(k+5),1);
        vector2 = abs(spectrogram(test2));
        vector2 = reshape(vector2,[length(vector2)*8,1]);
        B = [B vector2];
    end

end

test1 = B(:,1:25);
test2 = B(:,26:50);
test3 = B(:,51:75);

Test = [test1 test2 test3];
TestMat = U' * Test;  % PCA projection
pval = w' * TestMat;  % LDA projection

% Rock = 0, Jazz = 1, Classical = 2
ResVec = [];
for kk = 1:length(pval)
    if pval(kk) <= threshold1
        ResVec(kk) = 0;
    elseif threshold1 < pval(kk) <= threshold2
        ResVec(kk) = 1;
    else
        ResVec(kk) = 2;
    end
end
ResVec = ResVec';
hiddenlabels = [2*ones(25,1);zeros(25,1);ones(25,1)];
disp('Rate of success');
sucRate = 0;
for jj = 1:75
    if ResVec(jj) == hiddenlabels(jj)
        sucRate = sucRate + 1;
    end
end

```

```
sucRate = sucRate / 75

function [U,S,V,threshold1,threshold2,w,sorttype1,sorttype2,
    sorttype3] = music_trainer(train1_0,train2_0,train3_0)
n1 = size(train1_0,2);
n2 = size(train2_0,2);
n3 = size(train3_0,2);
A = [train1_0 train2_0 train3_0];
[U,S,V] = svd(A,'econ');
songs = S*V';
U = U(:,1:20);
type1 = songs(1:20, 1:n1);
type2 = songs(1:20, (n1+1):(n1+n2));
type3 = songs(1:20, (n1+n2+1):(n1+n2+n3));

m1 = mean(type1, 2);
m2 = mean(type2, 2);
m3 = mean(type3, 2);
m = (m1+m2+m3)/3;

Sw = 0; % within class variances
for k=1:n1
    Sw = Sw + (type1(:,k)-m1)*(type1(:,k)-m1)';
end
for k=1:n2
    Sw = Sw + (type2(:,k)-m2)*(type2(:,k)-m2)';
end
for k=1:n3
    Sw = Sw + (type3(:,k)-m3)*(type3(:,k)-m3)';
end

Sb = ((m1-m)*(m1-m)'+(m2-m)*(m2-m)'+(m3-m)*(m3-m)')/3; % between
    class

[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vtype1 = w'*type1;
vtype2 = w'*type2;
vtype3 = w'*type3;

mean(vtype1) % max
mean(vtype2) % min
mean(vtype3) % middle
```

```
sorttype1 = sort(vtype1);
sorttype2 = sort(vtype2);
sorttype3 = sort(vtype3);

t1 = length(sorttype2);
t2 = 1;
while sorttype2(t1)>sorttype3(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold1 = (sorttype2(t1)+sorttype3(t2))/2;

t3 = length(sorttype3);
t4 = 1;
while sorttype3(t3)>sorttype1(t4)
    t3 = t3 - 1;
    t4 = t4 + 1;
end
threshold2 = (sorttype3(t3)+sorttype1(t4))/2;

subplot(3,1,1)
histogram(sorttype1,50);hold on, plot([threshold1 threshold1], [0
    7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-1000 5000],'Ylim',[0 7])
title('classical')
subplot(3,1,2)
histogram(sorttype2,50);hold on, plot([threshold1 threshold1], [0
    7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-1000 5000],'Ylim',[0 7])
title('rock')
subplot(3,1,3)
histogram(sorttype3,50);hold on, plot([threshold1 threshold1], [0
    7],'r')
hold on, plot([threshold2 threshold2], [0 7],'g');
set(gca,'Xlim',[-1000 5000],'Ylim',[0 7])
title('jazz')

end
```