Linear Discriminant Analysis

By Dexian Chen

Abstract

The report intends to explore the application of Linear Discriminant Analysis (LDA) in analyzing signal data and machine learning for music classification. Generally, we read signal data from musical files and teach the computer with different data sets and then do a sample test to study the classifier algorithm.

Introduction and Overview

In 3 different tests, I prepare musical data sets for training and testing. The first test is to test accuracy of identifying bands from three different genres. The second test is to test accuracy of identifying different bands within the same genre. The third test is to test accuracy of identifying the genre, given various bands from three different genre.

Theoretical Background

Singular Value Decomposition (Equation 1) is a factorization as the following:

$$A = U\sum V^*$$

For any matrix A with a size of m × n, U is a m × m unitary matrix, $\sum$ is an m × n matrix whose diagonals are listing from largest to smallest and are non-negative real numbers called singular values denoted as σ, and V is a n × n unitary matrix. To apply PCA, we will perform SVD, and it will give us the dominant component in representing the data, which will help us determine the feature for **Linear Discriminant Analysis** (LDA). The idea of LDA is firstly proposed by Fisher in the context of taxonomy. The follow is the process for performing LDA:

For a two-class LDA, the above idea in results in consideration of the following mathematical formulation. Construct a projection **w** such that

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

, where the scatter matrices for between-class S_B and within-class S_W data are given by

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$
$$\mathbf{S}_W = \sum_{j=1}^{2} \sum_{\mathbf{x}} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T \, .$$

These quantities essentially measure the variance of the data sets as well as the variance of the difference in the means. Mathematical explanations are not necessary here to understand the musical classification, so this part will be skipped. After we build up **w**, we can find the eigenvalue by

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

The maximum eigenvalue λ and its associated eigenvector gives the quantity of interest and the projection basis. Once the scatter matrix is constructed, we're ready to perform LDA projection and form the decision threshold for music classifications.

Algorithm Implementation and Development

For each test, the algorithm is almost the same except for counting the number of errors according to the threshold, because the threshold will change in different data sets. Firstly, I locate my data file path in each test. Then I can iterate through all of the music files and read it into MATLAB. To prepare the data for LDA, I take the average my signal data and treat it as a mono type. Also, to make my data more manageable, I resample my data to a half with the **resample** command. Then for each song, I take several clips from it, with a length of 5 seconds. Simultaneously, I perform Fast Fourier Transform to transfer the absolute sound signal to frequency domain for analysis by **spectrogram** command. By reshaping those data horizontally in the **testmatrix**, I'm ready to train the data. Here, we write a **trainer** function (See Appendix B). The function primarily takes in the matrix we prepared and number of features. It returns U, S, V from the SVD decomposition, the scatter matrix **w**, which we explained in theoretical background, and the music groups matrix (**sortclassical, sortjazz, sortpop** in Appendix B) in a basis of **w**. By now, I train the data after the **trainer** is called. Next, I plot the three music groups in different horizontal lines to approximate the thresholds. To be more accurate, I plot the two thresholds by using the average value. After that, I repeat the process of preparing data when testing data. However, this time I perform LDA on the matrix direction by using the **U** and **w** from the trainer. Then we use a matrix **record** store the prediction. Since we know the "answer" of those test music file, we can create a **hiddenlabel**, compare it with our prediction matrix, and count the accuracy rate by figuring out the number of errors. (**See Appendix B Line88-100**). Eventually, we can finish training, testing and get an accuracy rate.

Computational Results

Test 1(Band Classification):

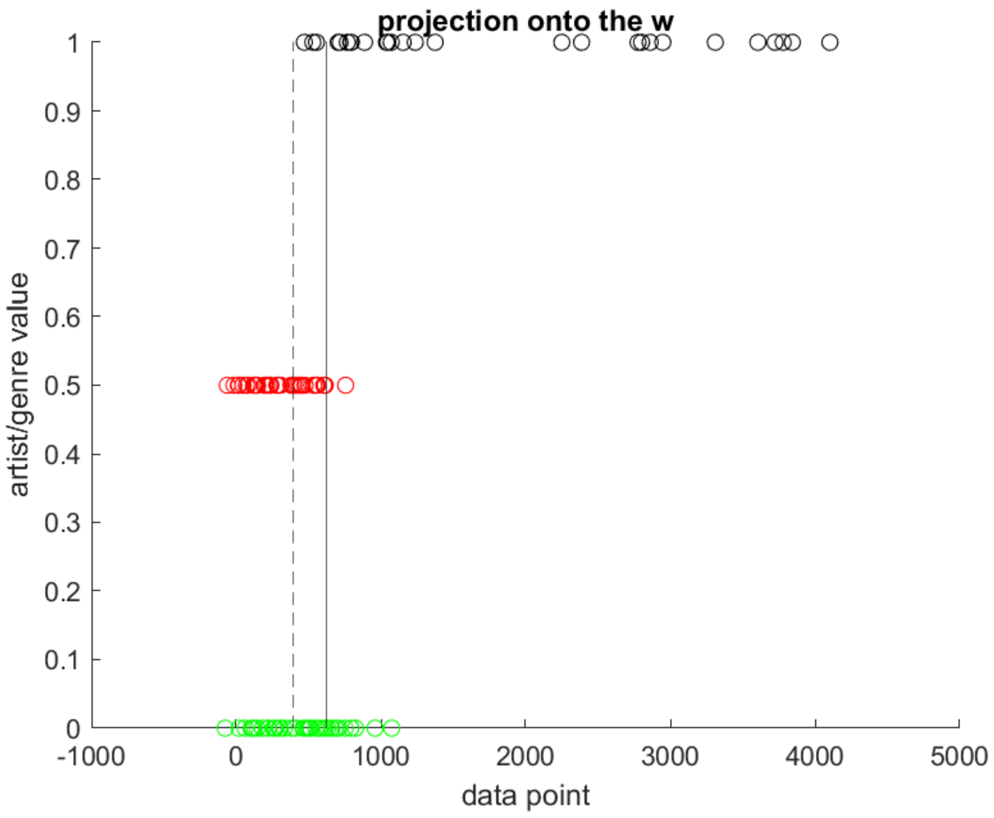

Figure 1: three different color represent three different bands

The artist in green is in Classical music genre. The artist in red is in Jazz genre. The artist in black is in pop genre. From Figure 1, we can see that the classifier does a good job classifying pop music but fail to classify Jazz and Classical very well. It has an accuracy of 45%. Reflecting on my music file which contains 3 songs from each band, I think the reason causing such low accuracy is a small size of samples, not diverse enough.

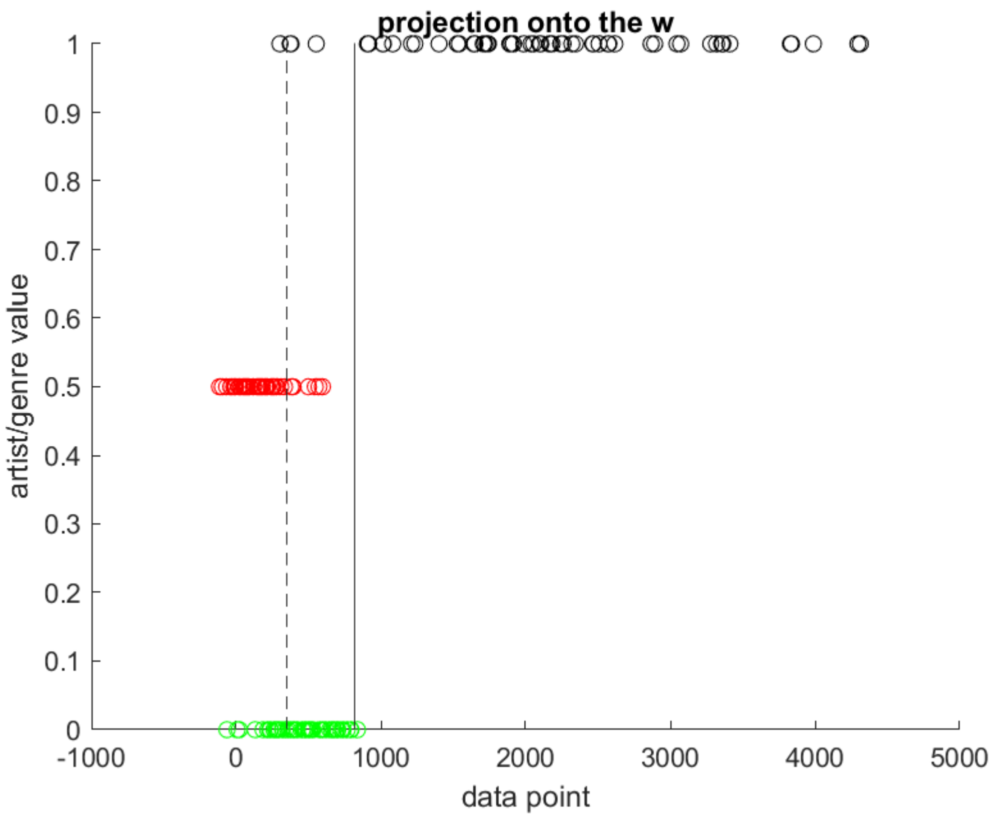

Figure 2: three different color represent three different bands in the same genre

Test 2(The case for Seattle):

The artists in green, red and black are three artists from the same genre. From Figure 2, we can tell that this a good approximation of separating three group because the overlap part of circles is smaller than Test1. It has an accuracy of 64%. However, this is not the original accuracy. This is improved by taking clips from different time range in a song. Hence, I know that the method I use has an disadvantage of getting samples. I need to take clips from different time range while increasing my sample size.

Test 3(Genre Classification):

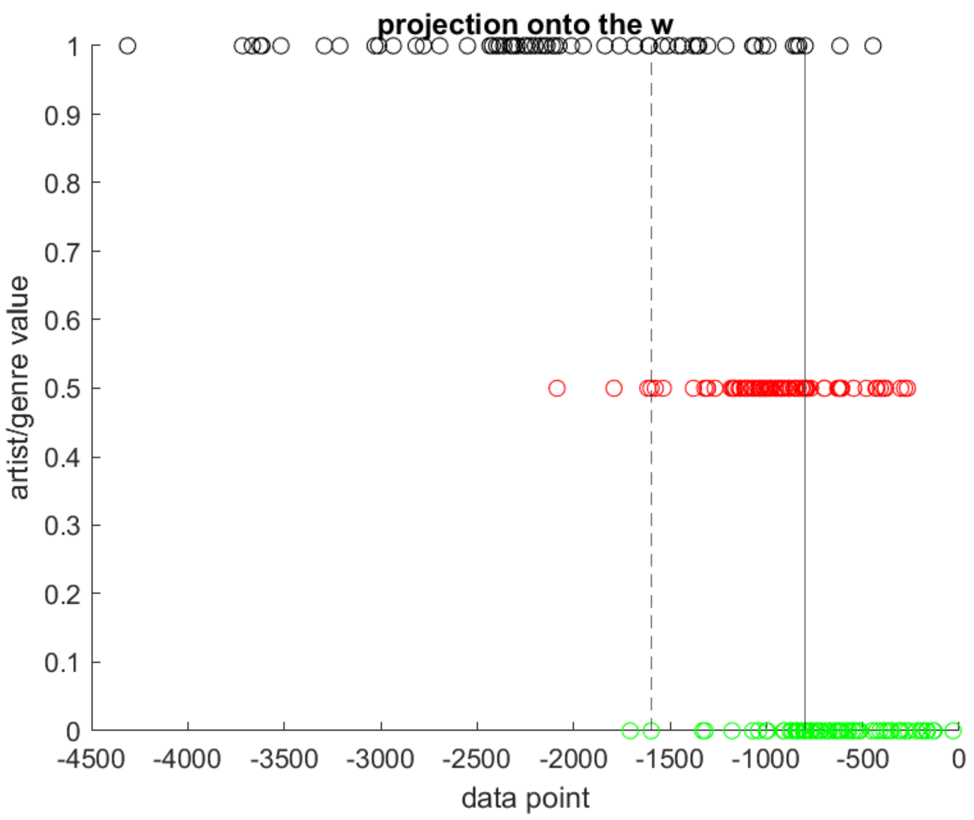

Figure 3: Green represent Classical music

Red represent Jazz music

Black represent Pop music

Different points represent different artists

From Figure 3, we can tell that the classifier does a fair predicting of genre classifications, but not good enough. It has an accuracy of 55%. Even though I triple my sample size and take less clips from each song to make my sample more diverse, I still cannot significantly increase my accuracy. From later research, I find that this is because many music can have more than one label. For example, some of my songs fall into both the category of Jazz music and Pop music. Since I'm not randomly choosing songs like a machine does, my subjective favor of music also influence the accuracy. Another improvement I can make is to do random sampling when choosing songs as training samples. Also, finding songs with non-overlapping genre labels would help increase the accuracy, too.

Summary and Conclusions

In summary, I load my chosen music files in MATLAB, and perform LDA to write my classifiers for classifying music in three tests. I'm able to accurately classify some groups but fails to classify the others. Reflecting on my process, there are some improvements that can be made during Machine Learning in music classification:

1. Random sampling songs to avoid personal favor of music.

2. Having a big and diverse enough sample size

3. Randomly choosing clips in each song

4. Avoid songs with overlap category label for classifying.

5. Improve Algorithm

Reference

Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data by J. Nathan Kutz.

Appendix A

Mean(X,2): return the input matrix X's mean in each row

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

Resample(x, p, q): resamples the input sequence, x, at p/q times the original sample rate. If x is a matrix, then resample treats each column of x as an independent channel. resample applies an antialiasing FIR lowpass filter to x and compensates for the delay introduced by the filter.

[V,D] = eig(A): returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that A*V = V*D

Appendix B

```
clear; close all; clc
% HW4
%% Case 1: Band Classification
%load music data
path = './song1/';
file = dir(path);
```

```
%iterate through each song
testmatrix = [];
for i = 3:length(file) %length(file) = 8
  filename = file(i).name;
  [y,Fs] = audioread(strcat(path, '/', filename));
  % Average the signal , treating it as mono type
  y = (y(:,1) + y(:,2))./2;
  y = y';
  % Cut 5 second clips as a sample in each song, y
  y = resample(y,Fs/2,Fs);
  Fs = Fs/2;
  for t = 20:5:90
      clip = y(1,Fs*t:Fs*(t+5));
      clipsp = abs(spectrogram(clip));
      clipfft = reshape(clipsp,[1,8*16385]);
      testmatrix = [testmatrix; clipfft];
  end
end
testmatrix = testmatrix';
feature = 20;
[U,S,V,w,sortclassical,sortjazz,sortpop] = trainer(testmatrix,feature);
figure(1)
hold on
nc = size(testmatrix,2)/3; nj = size(testmatrix,2)/3; np = size(testmatrix,2)/3;
plot(sortclassical, zeros(nc), 'go')
plot(sortjazz, 0.5*ones(nj), 'ro')
plot(sortpop, ones(np), 'ko')
title('projection onto the w')
ylabel('artist/genre values on projection w')
 xlabel('data point')
% From obervation of the graph, classical < threshold < jazz < pop
t1 = length(sortclassical);
t2 = 1;
while sortclassical(t1)>sortjazz(t2)
   t1 = t1-1;
   t2 = t2+1;
end
th1 = (sortclassical(t1)+sortjazz(t2))/2;
t3 = length(sortjazz);
t4 = 1;
while sortjazz(t3)>sortpop(t4)
   t3 = t3-1;
   t4 = t4+1;
end
th2 = (sortjazz(t3)+sortpop(t4))/2;
xline(th1,'--');
xline(th2, '.-');
% Test Classifier
%(1 classical, 1 jazz, 1 pop)
path = './test1/';
file = dir(path);
%iterate through each song
%(2 classical, 2 jazz, 2 pop) I may increase the number of songs downloaded
test = [];
for i = 3:length(file) %length(file) = 8
  filename = file(i).name;
  [y,Fs] = audioread(strcat(path, '/', filename));
  % Average the signal , treating it as mono type
  y = (y(:,1) + y(:,2))./2;
  y = y';
  % Cut 5 second clips as a sample in each song, y
  y = resample(y,Fs/2,Fs);
  Fs = Fs/2;
  for t = 20:5:70
     tclip = y(1,Fs*t:Fs*(t+5));
     tclipsp = abs(spectrogram(tclip));
     tclipfft = reshape(tclipsp,[1,8*16385]);
     test = [test; tclipfft];
  end
end
test = test';

nc2 = size(test,2)/3; nj2 = size(test,2)/3; np2 = size(test,2)/3;
classicallabel = zeros(1,nc2);
jazzlabel = ones(1,nj2)*0.5;
poplabel = ones(1,np2);
hiddenlabel = [classicallabel, jazzlabel, poplabel];
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection

totaltest = nc2 + nj2 + np2;
record = [];
for i = 1: totaltest
  if pval(i) <= th1
     record = [record;0];
  elseif pval(i) <= th2
     record = [record;0.5];
  else
     record = [record; 1];
  end
end
record = record';
```

```
errNum = sum(abs(record - hiddenlabel)~=0);
accuracy1 = 1 - errNum/totaltest

%% Case 2: The case for Seattle
%load music data
path = './song2/';
file = dir(path);
%iterate through each song
testmatrix = [];
for i = 3:length(file)
   filename = file(i).name;
   [y,Fs] = audioread(strcat(path, '/', filename));
   % Average the signal , treating it as mono type
   y = (y(:,1) + y(:,2))./2;
   y = y';
   % Cut 5 second clips as a sample in each song, y
   y = resample(y,Fs/2,Fs);
   Fs = Fs/2;
   for t = 10:5:100
       clip = y(1,Fs*t:Fs*(t+5));
       clipsp = abs(spectrogram(clip));
       clipfft = reshape(clipsp,[1,8*16385]);
       testmatrix = [testmatrix; clipfft];
   end
end
testmatrix = testmatrix';
figure(2)
feature = 20;
[U,S,V,w,sortclassical,sortjazz,sortpop] = trainer(testmatrix,feature);
% (green)classical here is Derek's music
% (red)Jazz here is Lobo's music
% (black)Pop here is Thorn's music
hold on
nc = size(testmatrix,2)/3; nj = size(testmatrix,2)/3; np = size(testmatrix,2)/3;
plot(sortclassical, zeros(nc), 'go')
plot(sortjazz, 0.5*ones(nj), 'ro')
plot(sortpop, ones(np), 'ko')
title('projection onto the w')
ylabel('artist/genre values on projection w')
 xlabel('data point')
% From obervation of the graph, classical < threshold < jazz < pop
t1 = length(sortjazz);
t2 = 1;
while sortjazz(t1)>sortclassical(t2)
    t1 = t1-1;
    t2 = t2+1;
```

```
end
th1 = (sortclassical(t1)+sortjazz(t2))/2;
t3 = length(sortclassical);
t4 = 1;
while sortclassical(t3)>sortpop(t4)
    t3 = t3-1;
    t4 = t4+1;
end
th2 = (sortclassical(t3)+sortpop(t4))/2;
xline(th1,'--');
xline(th2, '.-');
% Test Classifier
%(1 classical, 1 jazz, 1 pop)
path = './test2/';
file = dir(path);
%iterate through each song
test = [];
for i = 3:length(file)
   filename = file(i).name;
   [y,Fs] = audioread(strcat(path, '/', filename));
   % Average the signal , treating it as mono type
   y = (y(:,1) + y(:,2))./2;
   y = y';
   % Cut 5 second clips as a sample in each song, y
   y = resample(y,Fs/2,Fs);
   Fs = Fs/2;
   for t = 10:5:80
       tclip = y(1,Fs*t:Fs*(t+5));
       tclipsp = abs(spectrogram(tclip));
       tclipfft = reshape(tclipsp,[1,8*16385]);
       test = [test; tclipfft];
   end
end
test = test';

nc2 = size(test,2)/3; nj2 = size(test,2)/3; np2 = size(test,2)/3;
classicallabel = zeros(1,nc2);
jazzlabel = ones(1,nj2)*0.5;
poplabel = ones(1,np2);
hiddenlabel = [classicallabel, jazzlabel, poplabel];
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection

totaltest = nc2 + nj2 + np2;
record = [];
for i = 1: totaltest
```

```
    if pval(i) <= th1
        record = [record;0.5];
    elseif pval(i) <= th2
        record = [record;0];
    else
        record = [record; 1];
    end
end
record = record';
errNum = sum(abs(record - hiddenlabel)~=0);
accuracy2 = 1 - errNum/totaltest
%% Case 3: Genre Classification
%load music data
path = './song3/';
file = dir(path);
%iterate through each song
testmatrix = [];
for i = 3:length(file)
    filename = file(i).name;
    [y,Fs] = audioread(strcat(path, '/', filename));
    % Average the signal , treating it as mono type
    y = (y(:,1) + y(:,2))./2;
    y = y';
    % Cut 5 second clips as a sample in each song, y
    y = resample(y,Fs/2,Fs);
    Fs = Fs/2;
    for t = 30:5:70
        clip = y(1,Fs*t:Fs*(t+5));
        clipsp = abs(spectrogram(clip));
        clipfft = reshape(clipsp,[1,8*16385]);
        testmatrix = [testmatrix; clipfft];
    end
end
testmatrix = testmatrix';
figure(3)
feature = 20;
[U,S,V,w,sortclassical,sortjazz,sortpop] = trainer(testmatrix,feature);
% (green)classical here is Derek's music
% (red)Jazz here is Lobo's music
% (black)Pop here is Thorn's music
hold on
nc = size(testmatrix,2)/3; nj = size(testmatrix,2)/3; np = size(testmatrix,2)/3;
plot(sortclassical, zeros(nc), 'go')
plot(sortjazz, 0.5*ones(nj), 'ro')
plot(sortpop, ones(np), 'ko')
title('projection onto the w')
```

```
ylabel('artist/genre values on projection w')
 xlabel('data point')
% From obervation of the graph, classical < threshold < jazz < pop
t1 = length(sortpop);
t2 = 1;
while sortpop(t1)>sortjazz(t2)
    t1 = t1-1;
    t2 = t2+1;
end
th1 = (sortjazz(t1)+sortpop(t2))/2;
t3 = length(sortjazz);
t4 = 1;
while sortjazz(t3)>sortclassical(t4)
    t3 = t3-1;
    t4 = t4+1;
end
th2 = (sortclassical(t3)+sortjazz(t4))/2;
xline(th1,'--');
xline(th2, '.-');
% Test Classifier
%(1 classical, 1 jazz, 1 pop)
path = './test3/';
file = dir(path);
%iterate through each song
%(2 classical, 2 jazz, 2 pop) I may increase the number of songs downloaded
test = [];
for i = 3:length(file) %length(file) = 8
   filename = file(i).name;
   [y,Fs] = audioread(strcat(path, '/', filename));
   % Average the signal , treating it as mono type
   y = (y(:,1) + y(:,2))./2;
   y = y';
   % Cut 5 second clips as a sample in each song, y
   y = resample(y,Fs/2,Fs);
   Fs = Fs/2;
   for t = 10:5:40
       tclip = y(1,Fs*t:Fs*(t+5));
       tclipsp = abs(spectrogram(tclip));
       tclipfft = reshape(tclipsp,[1,8*16385]);
       test = [test; tclipfft];
   end
end
test = test';

nc2 = size(test,2)/3; nj2 = size(test,2)/3; np2 = size(test,2)/3;
classicallabel = zeros(1,nc2);
```

```
jazzlabel = ones(1,nj2)*0.5;
poplabel = ones(1,np2);
hiddenlabel = [classicallabel, jazzlabel, poplabel];
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection

totaltest = nc2 + nj2 + np2;
record = [];
for i = 1: totaltest
   if pval(i) <= th1
       record = [record;1];
   elseif pval(i) <= th2
       record = [record;0.5];
   else
       record = [record; 0];
   end
end
record = record';
errNum = sum(abs(record - hiddenlabel)~=0);
accuracy3 = 1 - errNum/totaltest
%%
function [U,S,V, w,sortclassical,sortjazz,sortpop] = trainer(testmatrix,feature)
    nc = size(testmatrix,2)/3; nj = size(testmatrix,2)/3; np = size(testmatrix,2)/3;

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

    artist = S*V'; % projection onto principal components
    U = U(:,1:feature);
    classical = artist(1:feature,1:nc);
    jazz = artist(1:feature,nc+1:nc+nj);
    pop = artist(1:feature,nc+nj+1:nc+nj+np);

    mc = mean(classical,2);
    mj = mean(jazz,2);
    mp = mean(pop,2);
    totalm = mean(artist(1:feature),2);

    Sw = 0; % within class variances
    for k=1:nc
        Sw = Sw + (classical(:,k)-mc).*(classical(:,k)-mc)';
    end
    for k=1:nj
        Sw = Sw + (jazz(:,k)-mj).*(jazz(:,k)-mj)';
    end
    for k=1:np
       Sw = Sw + (pop(:,k)-mp).*(pop(:,k)-mp);
    end

    Sb = (mc-totalm)*(mc-totalm)'+(mj-totalm)*(mj-totalm)'+(mp-totalm)*(mp-totalm)'; %
between class

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

    vclassical = w'*classical; % projection onto the threshold
    vjazz = w'*jazz;
    vpop = w'*pop;

    sortclassical = sort(vclassical);
    sortjazz = sort(vjazz);
    sortpop = sort(vpop);
end
```