Music Classification with Linear Discrimination Analysis

Yuxin Jiang
AMATH 482 HW4

Abstract

In this report, we will examine different clips of music to classify them into different artists, or different genres, using SVD decomposition and Linear Discrimination Analysis (LDA). We will decompose spectrogram of music clips, which gave us wavelet basis functions. With the decomposed data, we will find the principal components and design thresholds for discrimination between different artists or different music genres. In the end, we will run our classifier on a test set, which are new clips of music, to test the success rate (accuracy) for a better perception of LDA and classifier.

I. Introduction and Overview

LDA is one of the most effective methods for statistical decisions. In this experiment, we will first decompose the music clips datasets using SVD for principal components or proper orthogonal mode decompositions that will represent our music clips. Then we will apply LDA algorithm which will provide us with a threshold that helps us identify music clips with different standards. We are going to do three tests to test the accuracy of different classifiers. In the first test, we will choose two songs from each different artist as our training set, taking around 15 5-second clips from each song, and build an artist (band) classifier. We will then use several 5-second clips from a new song of each artist as our test sets and calculate the success rate. I choose songs from Chopin, AC/DC and Justin Bieber in this case. In the second test, we will repeat the previous steps, but we will be using songs of 3 different artists from Seattle in the same genre to further test our accuracy. I choose Soundgarden, Alice in Chains and Pearl Jam. In the last test, we will use the same algorithm but with songs of different artists from 3 different genres - pop, classical and rock - to test the accuracy of a genre classifier.

II. Theoretical Background

A. Singular Value Decomposition (SVD)

A Singular Value Decomposition (SVD) is a factorization that converts a matrix of size $m \times n$ as following:

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

where $U \in \mathbb{C}^{m\times n}$, $V \in \mathbb{C}^{n\times n}$, $\Sigma \in \mathbb{R}^{m\times n}$. It is also assumed that the diagonal entries of Σ are nonnegative and ordered from largest to smallest so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p = \min(m, n)$.

With SVD, the matrix first applies a unitary transformation preserving the unit sphere with V^*. Then it is stretched by Σ that creates an ellipse with principal semiaxes. In the end, the generated hyperellipse is rotated by the unitary transformation U. Every matrix has an SVD and singular values σ_j are distinct, which enables us to diagonalize every matrix.

B. 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 multiple-class LDA, we will construct a projection w such that

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

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

$$S_B = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T \tag{3}$$

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

where μ_i and μ_j are the mean of each class, C is the number of classes, and μ is the mean of the class means.

The criterion is commonly known as the generalized Rayleigh quotient whose solution can be found via the generalized eigenvalue problem

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

where the maximum eigenvalue and its associated eigenvector give a quantity of interest and the projection basis. With LDA, data are well separated with the means being well apart when projected onto a chosen subspace. [1]

III. Algorithm Implementation and Development

In the three tests, I used similar algorithm with changes in number of samples and order of classes in LDA to build several classifiers. In the end, I set up test sets and compared to the classification results of the classifiers I built for the accuracy rate.

In the first test where we need to build an artist classifier, I chose two songs from each of Chopin, AC/DC and Justin Bieber, artists from different music genres, as the training set, and chose one new song from the same artists as the test set. I used a for loop and the command **audioread** to load the songs that gives us the data and number of data points per second. Since the songs are all

[1] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

loaded as ‘stereo’, which contains two columns of data, I transferred them into ‘mono’ by taking the average of two columns. Then I used a for loop to take 19 5-second music pieces from every song to ensure better results. I used **Fs*j** to convert the seconds to number of data. Within each iteration, I also took the spectrogram of the data using **spectrogram** so that dominant spectrogram modes associated will be given and we do not need to perform wavelet transforms again. I reshaped the data from 32769*8 to 1*(32769*8) and combined data of different songs into a training matrix. I then took the transpose of the training matrix and calculated the length of data from 3 artists by dividing the number of columns in the training matrix by 3. I then applied SVD to the training matrix and projected the data onto principal components by multiplying the transpose of V and S. I chose the feature as 20 since I wished to project onto 20 principal components in total. I then separated the data in the way I combined them in the beginning and calculated the mean of each class with the command **mean**, as well as the general mean **m**. I then calculated the within class variance using a for loop and scatter matrices for between class according to equation (3) and (4). I then applied the LDA according to equation (2) and (5) to find w. Then we could be able to get the projection through **w’*chopin**. In order to find the threshold, I first examined the mean of each projected vector. I ranked them from the least to the largest and aimed at finding the two thresholds between each class. I first sorted the data of the vector of each class in ascending order. I started by comparing the largest element of the smaller-ranked class with the smallest element of the larger-ranked class and find a threshold where two classes can be distinguished. I took the average of the thresholds in two classes as the final threshold for determination of different songs. For example, in test 1, I found that mean of each vector is ranked from the smallest to the largest as AC/DC, Chopin and Justin Bieber. Therefore, I inserted two thresholds between AC/DC and Chopin, and between Chopin and Justin Bieber. By comparing the largest of AC/DC with smallest of Chopin, I found the balanced indexes as I moved to smaller number in AC/DC and larger number in Chopin. By taking the average of data according to the balanced indexes, I will be able to determine the corresponding thresholds.

After finding out the thresholds, I started setting up the testing set. Following similar steps in the beginning, I loaded the data in a for loop using **audioread**, took 11 pieces of music from every song, took the spectrogram and reshaped the data. Then in order to create a predicted value using the classifier, I first applied the PCA projection by **U’*test**. Then I applied LDA projection similar as steps used to build the classifier, **w’*TestMat** and saved the value in **pval**. In order to visualize our results, I plotted the predicted results and thresholds onto a histogram using **bar**. In order to calculate the accuracy rate, I used different numbers to represent different artists: AC/DC as -1, Justin Bieber as 1, and Chopin as 0. I built a for loop for the 33 test pieces, and using **if** to change the test data into numeric representations of the artists, saving them in vector **p**. I also set **truep** with corresponding numeric representations in the order I constructed the test sets, that is Chopin, AC/DC and Justin Bieber. By subtracting **p** by **truep**, I am able to see the number of successes by finding number of zeros existing in **res**. Since I took 11 pieces of music from 3 different songs,

the sample size is 33. Through dividing the number of successes (the number of zeros) by the sample size 33, I am able to get the accuracy as a result.

I did the similar for the rest two tests but with different choice of songs and different ranking while finding the thresholds. In test 2, I chose two songs from each of Soundgarden, Alice in Chains and Pearl Jam, who are all rock artists from Seattle, as the training set for band classification, and one new song from each artist as the testing set. I chose 14 pieces in each song in training set and 11 pieces in each song in testing sets. In test 3, I chose two songs from each of the genre including pop, classical and rock music as training set for genre classification, and one new song in each genre as testing set. I chose songs from Ariana Grande, Justin Bieber and Sam Smith for pop music, Chopin, Beethoven and Liszt for classical music, and AC/DC, Led Zeppelin and Deep Purple for rock music. Among them, I chose 15 pieces for training set and 11 pieces for testing set for each song. In the end, I found the accuracy rates in different scenarios using the same way as described above.

IV. Computational Results

A. Test 1: Band Classification

The accuracy rate in test 1 is 96.97%. As we can see in figure (1), the classifier we built using LDA could be able to accurately identify music from different artists in the most cases. The thresholds are relatively accurate comparing to the actual result. There are small number of deviations, but the success rate is relatively high.

B. Test 2: Seattle band

The accuracy rate in test 2 is around 54.55%. Looking at figure (2), it is not hard to see that the success rate is much lower than that in the first case. Since the music we chose are from

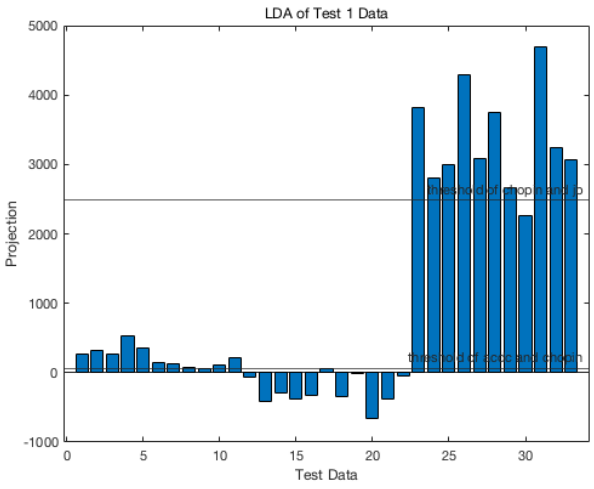

Figure (1): Histogram of Results on Test Data in Test 1 with Thresholds

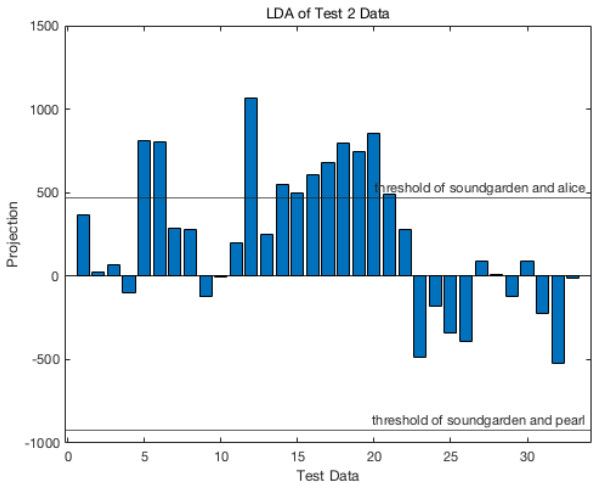

Figure (2): Histogram of Results on Test Data in Test 2 with Thresholds

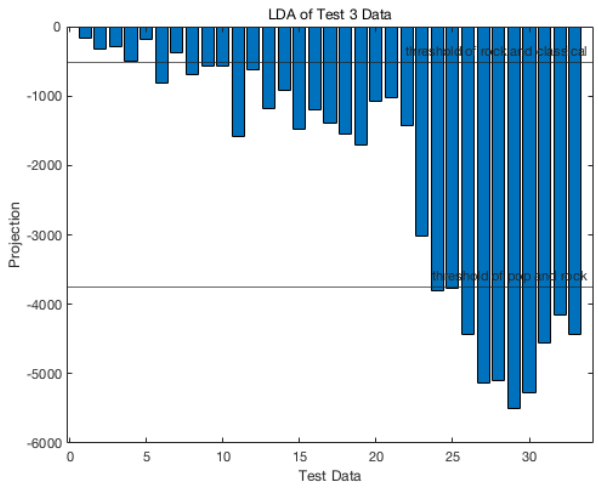

Figure (3): Histogram of Results on Test Data in Test 3 with Thresholds

the same genre, it is harder for the classifier to identify different artists (bands) since their music styles are similar.

C. Test 3: Genre classification

The accuracy rate in test 3 is 81.82%. This success rate is higher than in test 2, but lower than test 1. The classifier can still offer us great amount of information given the success rate. However, the results may also due to the choice of music I selected, which are mostly typical songs in each genre, thus making it easier for classifier to identify the songs.

V. Summary and Conclusions

With this experiment, we could clearly see the usefulness of LDA and SVD in music classification. In test 1, the accuracy rate is really high when classifying songs from different genres and different artists when the training set is composed of similar songs from each artist. However, by comparing with test 2, it is easy to see that the accuracy rate decreases rapidly while identifying songs from different artists but the same genre. The success rate is highly influenced by the degree of difference between different classes during classification. In test 3, the success rate is smaller than that of test 1 since we are using different artists in training set. By comparing the success rates in test 3 and test 1, we can also conclude that the accuracy is influenced by the degree of similarity between songs of specific class in the training set. Moreover, the number of samples in training sets can also contribute to the accuracy since test 1 contains the greatest number of samples in training set, 19, and test 2 contains the least number of samples, 14. Therefore, in order to build a classifier with high accuracy, we need to add as many as possible samples, and use songs that are highly typical with appropriate similarity and difference while building the training set.

Appendix A: MATLAB Functions Used and Brief Implementation Explanation

[U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'.

[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.

[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

bar(y) creates a bar graph with one bar for each element in y. If y is an m-by-n matrix, then bar creates m groups of n bars.

B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B).

B = sort(A) sorts the elements of A in ascending order.

M = mean(A) returns the mean of the elements of A along the first array dimension whose size does not equal 1.

s = num2str(A) converts a numeric array into a character array that represents the numbers.

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.

s = strcat(s1,...,sN) horizontally concatenates s1,...,sN. Each input argument can be a character array, a cell array of character vectors, or a string array.[2]

[2] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/

Appendix B: MATLAB Code

```
clear all; close all; clc;

%% band classification
train=[];
for str=["chopin","acdc","jb"]
    for i=1:2
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        song=song'/2;
        song=song(1,:)+song(2,:);
        for j=20:10:200 % 19 piece from every music for better results
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            train=[train; fivespec];
        end
    end
end
train=train';
sz = size(train,2)/3;

[U,S,V] = svd(train,'econ');
feature=20;
music = S*V'; % projection onto principal components
U = U(:,1:feature);
chopin = music(1:feature,1:sz);
acdc = music(1:feature,sz+1:2*sz);
jb=music(1:feature, 2*sz+1:3*sz);

ma = mean(chopin,2);
md = mean(acdc,2);
mc = mean(jb,2);
m = (ma+md+mc)/3;

    Sw = 0; % within class variances
    for k=1:sz
        Sw = Sw + (chopin(:,k)-ma)*(chopin(:,k)-ma)';
    end
    for k=1:sz
        Sw = Sw + (acdc(:,k)-md)*(acdc(:,k)-md)';
    end
    for k=1:sz
        Sw = Sw + (jb(:,k)-mc)*(jb(:,k)-mc)';
    end

    Sb = ((ma-m)*(ma-m)'+(md-m)*(md-m)'+(mc-m)*(mc-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);

    vchopin = w'*chopin;
    vacdc = w'*acdc;
    vjb= w'*jb;
```

```
% acdc < threshold < chopin < threshold < jb

    sortchopin = sort(vchopin);
    sortacdc = sort(vacdc);
    sortjb=sort(vjb);

    t1 = length(sortacdc);
    t2 = 1;
    while sortacdc(t1)>sortchopin(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    thresholdacdcchopin = (sortacdc(t1)+sortchopin(t2))/2;

    t3=length(sortchopin);
    t4=1;
    while sortchopin(t3)>sortjb(t4)
        t3=t3-1;
        t4=t4+1;
    end
    thresholdchopinjb = (sortchopin(t3)+sortjb(t4))/2;


%test1
test=[];
for i=3
    for str=["chopin","acdc","jb"]
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        song=song'/2;
        song=song(1,:)+song(2,:);
        for j=20:10:120 % 11 piece from every music
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            test=[test; fivespec];
        end
    end
end


test=test.';
TestNum=size(test,2);
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection
figure(1)
bar(pval)
xlabel('Test Data')
ylabel('Projection')
title('LDA of Test 1 Data')
hold on;
yline(thresholdacdcchopin,'-','threshold of acdc and chopin');
hold on;
```

```
yline(thresholdchopinjb,'-','threshold of chopin and jb');
hold off;

p=pval;
for i=1:33
    if p(i)<thresholdacdcchopin
        p(i)=-1;%acdc
    elseif  p(i)>thresholdchopinjb
        p(i)=1;%jb
    else thresholdacdcchopin<p(i)<thresholdchopinjb
        p(i)=0;%chopin
    end
end

truep=zeros(1,33);
truep(12:22)=-1;
truep(23:33)=1;

res=p-truep;
k=0;
for i=1:33
    if res(i)==0
        k=k+1;
    end
end

suc1=k/33;

%% the case of seattle
train=[];
for str=["soundgarden","alice","pearl"]
    for i=1:2
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        song=song'/2;
        song=song(1,:)+song(2,:);
        for j=10:10:140 % 14 piece from every music for better results
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            train=[train; fivespec];
        end
    end
end
train=train';
sz = size(train,2)/3;

[U,S,V] = svd(train,'econ');
feature=20;
music = S*V'; % projection onto principal components
U = U(:,1:feature);
soundgarden = music(1:feature,1:sz);
alice = music(1:feature,sz+1:2*sz);
pearl=music(1:feature, 2*sz+1:3*sz);

ma = mean(soundgarden,2);
```

```
md = mean(alice,2);
mc = mean(pearl,2);
m = (ma+md+mc)/3;

    Sw = 0; % within class variances
    for k=1:sz
        Sw = Sw + (soundgarden(:,k)-ma)*(soundgarden(:,k)-ma)';
    end
    for k=1:sz
        Sw = Sw + (alice(:,k)-md)*(alice(:,k)-md)';
    end
    for k=1:sz
        Sw = Sw + (pearl(:,k)-mc)*(pearl(:,k)-mc)';
    end

    Sb = ((ma-m)*(ma-m)'+(md-m)*(md-m)'+(mc-m)*(mc-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);

    vsoundgarden = w'*soundgarden;
    valice = w'*alice;
    vpearl= w'*pearl;

% pearl < threshold < soundgarden < threshold < alice

    sortsoundgarden = sort(vsoundgarden);
    sortalice = sort(valice);
    sortpearl=sort(vpearl);

    t1 = length(sortpearl);
    t2 = 1;
    while sortpearl(t1)>sortsoundgarden(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    thresholdpearlsoundgarden = (sortpearl(t1)+sortsoundgarden(t2))/2;

    t3=length(sortsoundgarden);
    t4=1;
    while sortsoundgarden(t3)>sortalice(t4);
        t3=t3-1;
        t4=t4+1;
    end
    thresholdsoundgardenalice = (sortsoundgarden(t3)+sortalice(t4))/2;


%test1
test=[];
for i=3
    for str=["soundgarden","alice","pearl"]
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        song=song'/2;
```

```
        song=song(1,:)+song(2,:);
        for j=20:10:120 % 11 piece from every music
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            test=[test; fivespec];
        end
    end
end

test=test.';
TestNum=size(test,2);
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection
figure(2)
bar(pval)
xlabel('Test Data')
ylabel('Projection')
title('LDA of Test 2 Data')
hold on;
yline(thresholdpearlsoundgarden,'-','threshold of soundgarden and pearl');
hold on;
yline(thresholdsoundgardenalice,'-','threshold of soundgarden and alice');
hold off;
p=pval;
for i=1:33
    if p(i)<thresholdpearlsoundgarden
        p(i)=-1;%pearl
    elseif  p(i)>thresholdsoundgardenalice
        p(i)=1;%alice
    else thresholdpearlsoundgarden<p(i)<thresholdsoundgardenalice
        p(i)=0;%soundgarden
    end
end

truep=zeros(1,33);
truep(12:22)=1;
truep(23:33)=-1;

res=p-truep;
k=0;
for i=1:33
    if res(i)==0
        k=k+1;
    end
end

suc2=k/33;


%% genre classification
train=[];
for str=["classical","rock","pop"]
    for i=1:2
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
```

```
        song=song'/2;
        song=song(1,:)+song(2,:);
        for j=10:10:150 % 15 piece from every music for better results
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            train=[train; fivespec];
        end
    end
end
train=train';
sz = size(train,2)/3;

[U,S,V] = svd(train,'econ');
feature=20;
music = S*V'; % projection onto principal components
U = U(:,1:feature);
classical = music(1:feature,1:sz);
rock = music(1:feature,sz+1:2*sz);
pop=music(1:feature, 2*sz+1:3*sz);

ma = mean(classical,2);
md = mean(rock,2);
mc = mean(pop,2);
m = (ma+md+mc)/3;

    Sw = 0; % within class variances
    for k=1:sz
        Sw = Sw + (classical(:,k)-ma)*(classical(:,k)-ma)';
    end
    for k=1:sz
        Sw = Sw + (rock(:,k)-md)*(rock(:,k)-md)';
    end
    for k=1:sz
        Sw = Sw + (pop(:,k)-mc)*(pop(:,k)-mc)';
    end

    Sb = ((ma-m)*(ma-m)'+(md-m)*(md-m)'+(mc-m)*(mc-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);

    vclassical = w'*classical;
    vrock = w'*rock;
    vpop= w'*pop;

% pop < threshold < rock < threshold < classical

    sortclassical = sort(vclassical);
    sortrock = sort(vrock);
    sortpop=sort(vpop);

    t1 = length(sortpop);
```

```
    t2 = 1;
    while sortpop(t1)>sortrock(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    thresholdpoprock = (sortpop(t1)+sortrock(t2))/2;

    t3=length(sortrock);
    t4=1;
    while sortrock(t3)>sortclassical(t4);
        t3=t3-1;
        t4=t4+1;
    end
    thresholdrockclassical = (sortrock(t3)+sortclassical(t4))/2;


%test1
test=[];
for i=3
    for str=["classical","rock","pop"]
        [song, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        song=song'/2;
        song=song(1,:)+song(2,:);
        for j=20:10:120 % 11 piece from every music
            five=song(Fs*j:Fs*(j+5));
            fivespec=abs(spectrogram(five));
            fivespec=reshape(fivespec, [1,32769*8]);
            test=[test; fivespec];
        end
    end
end

test=test.';
TestNum=size(test,2);
TestMat = U'*test;  % PCA projection
pval = w'*TestMat;  % LDA projection
figure(3)
bar(pval)
xlabel('Test Data')
ylabel('Projection')
title('LDA of Test 3 Data')
hold on;
yline(thresholdpoprock,'-','threshold of pop and rock');
hold on;
yline(thresholdrockclassical,'-','threshold of rock and classical');
hold off;
p=pval;
for i=1:33
    if p(i)<thresholdpoprock
        p(i)=-1;%pop
    elseif  p(i)>thresholdrockclassical
        p(i)=1;%classical
    else thresholdpoprock<p(i)<thresholdrockclassical
        p(i)=0;%rock
    end
```

```
end

truep=zeros(1,33);
truep(1:11)=1;
truep(23:33)=-1;

res=p-truep;
k=0;
for i=1:33
    if res(i)==0
        k=k+1;
    end
end

suc3=k/33;
```