Music Classification

Chenghao Chen
AMATH 482 HW4

Abstract

In this homework, we will be working on music. Music usually have different genres and are produced by different artists. Different artists have their different style and genre in their songs. Therefore, we want to examine several clips of songs to classify into different artists or different genres by using Linear Discrimination Analysis. More importantly, we want to use our classifier to test the accuracy in different situations.

Introduction

In the homework, we want to transfer song clips to the spectrogram and then decompose the music clips using SVD. Then, we can work on Linear Discrimination Analysis, which is used for statistical decision. With the help of LDA algorithm, we can identify thresholds of the system and use it as a classifier. There are three parts of the project. The first part is to test different artists with different genre using a classifier. The second one is to test different artist with same genre using a classifier. The third one is to test different genre with different artists using a classifier. At last, we calculate the accuracy of three classifier.

Theoretical Background

Linear Discrimination Analysis

Consider a set of observations (also called features, attributes, variables or measurements) for each sample of an object or event with known class. This set of samples is called the training set. The classification problem is then to find a good predictor for the class of any sample of the same distribution (not necessarily from the training set) given only an observation x.

In the case where there are more than two classes, the analysis used in the derivation of the Fisher discriminant can be extended to find a subspace which appears to contain all of the class variability. Suppose that each of C classes has a mean and the same covariance. Then the scatter between class variability may be defined by the sample covariance of the class means

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

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

If SB is diagonalizable, the variability between features will be contained in the subspace spanned by the eigenvectors corresponding to the C − 1 largest eigenvalues. These eigenvectors are primarily used in feature reduction, as in PCA. The eigenvectors corresponding to the smaller eigenvalues will tend to be very sensitive to the exact choice of training data, and it is often necessary to use regularisation as described in the next section. [1]

SVD

A singular value decomposition (SVD) is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications. The SVD, much as illustrated in the preceding paragraph, is essentially a transformation that stretches/compresses and rotates a given set of vectors.

$$A = U\Sigma V^* \tag{3}$$

Algorithm Implementation

In this homework, we have three different tests to do, but they are all pretty similar. Therefore, I create three functions that can be used in three tests. Then, I calculate the accuracy at the end.

The first function is to load the song.

```
                              function [song0] = loading(name)                    (4)
```

First, I use a for loop to load all the songs into my computer by command **audioread**, which can give us data and frequency. Then, I take the average of the data because the songs are all stereo, which has two columns of data. Then, I use a for loop to extract 21 pieces of five second music clips from the song. Also, I took the spectrogram of the music clips and reshape the data to make it [1 * (32769 * 8*)]. Finally, I store these 21 pieces in a new vector called song0.

The second function is to make a classifier.

```
function [U,S,V,threshold1, threshold2,w,sortpop,sortrock,sortclassic,a] =
dc_trainer(pop0,rock0,classic0,feature)                                         (5)
```

The function takes three different types of music as parameters, pop0, rock0 and classic0. The parameter feature is the number of principal components I want to project in total. First, I get the sizes of three genres using command **size.** Then, I use SVD to three genres and project the data onto principal components by multiplying the transpose of V and S. Then, I want to separate three genres by choosing different indexes in the data. I also want to get the mean of each genre. Then, I want to calculate within class variance by using three while loops and calculate between class variance. Then, I can use LDA to calculate w and we find the projection of three genres and sort them. Now, we want to find two thresholds because we have three genres. There are six situations in this way.

```
First situation   pop < threshold1 < rock < threshold2 < classic
Second situation  pop < threshold1 < classic < threshold2 < rock
Thrid situation   classic < threshold1 < pop < threshold2 < rock
Fourth situation  rock < threshold1 < pop < threshold2 < classic
Fifth situation   rock < threshold1 < classic < threshold2 < pop
Sixth situation   classic < threshold1 < rock < threshold2 < pop
```

I will use a big if loop with several small if loops to identify different situations. In the big if loop, it is either pop > rock or pop < rock. Then, in each situation, there are three different situations. I have list my code to help you better understand. In each situation, I find the average between least and median and the average between median and largest. These two averages are two threshold we want. At last, the function will return U S V from SVD, two thresholds, sorted three genres, and the number of situations in six situations.

```
if mean(vpop) <= mean(vrock)
        if mean(vrock) <= mean(vclassic)
    first situation
        else
```

```
            if mean(vclassic) > mean(vpop)
      second situation
            else
      third situation
            end
        end
    else
        if mean(vpop) <= mean(vclassic)
      fourth situation
        else
            if mean(vclassic) > mean(vrock)
      fifth situation
            else
      sixth situation
            end
        end
```

The third function is to classify the test.

```
function [pval] = classify(songname,U,w)                                  [6]
```

First, I still need to load the music I want to test and extract 21 pieces of five seconds music clips using the same method I use in first function. Then, I will have the spectrogram of the music clips. Then, I will multiply U' for PCA projection and w' for LDA projection. At last, we can compare the values with two thresholds.

With these three main functions, my main code is really easy to understand. In the first test, we want to test three different artists, Ariana Grande(pop), Pink Floyd(rock), and Chopin(classic). First, I load three artists two songs, each with 21 pieces of five seconds music clips, into the system. Then, I create a classifier that can identify three artists. At last, we can test the classifier by testing a new song for each artist.

In the second test, we want to test three different artists with same genre, Soundgarden, Alice in Chains and Pearl Jam. First, I load three artists two songs, each with 21 pieces of five seconds music clips, into the system. Then, I create a classifier that can identify three artists. At last, we can test the classifier by testing a new song for each artist.

In the third test, we want to test three different genres with different artists, (Ariana, justin for pop), (chopin and Mozart for classic), (ACDC and Pink Floyd for rock).First, I load three artists two songs, each with 21 pieces of five seconds music clips, into the system. Then, I create a classifier that can identify three artists. At last, we can test the classifier by testing a new song for each artist.

Computational Results

Test 1

a = 6 Situation 6: `classic < threshold1 < rock < threshold2 < pop`
threshold1 = -57.1547
threshold2 =1.3308e+03
chopin < -57.15 < Pink < 1330.8 < Ariana

Pink test:

125.47	37.54	51.72	92.33	-1.66	299.63	201.44	247.7	593.5	42.88	104.95
130.97	-4.5	4.9	18.32	-5.5	23.64	4.27	47.7	-115	540.5	

Table[1]

Accuracy: 20/21
Ariana:

2470	1977	4574	5324	4172	5433	4858	5128	5031	5380	2444
2491	2388	2905	2345	2754	2841	5154	5396	5311	5047	

Table[2]

Accuracy: 21/21
Chopin:

-181.9	-262.8	-79.6	-155.8	-152.4	-292.2	-262.5	-270.6	-163.2	-194.8	-178.4
-128.2	-129.4	-56.4	-126.8	-208.8	-374.5	-80.3	-82.5	-106.8	-225.1	

Table[3]

Accuracy: 20/21
Total accuracy: 61/63 = 96.8%
As we can see, the classifier we build can successfully identify different artists with different genres.

<u>Test 2</u>
a = 2 Situation 2: `pop < threshold1 < classic < threshold2 < rock`

threshold1 = -277.2819
threshold2 = 2.3757e+03
alice < -277.28 < Soundgarden < 2375.7 < pearl

alice test

-678.7	-240	65.98	-230.2	-249.4	-232.2	-285.7	-31.85	-317	-704.7	-410.7
-693.7	-518	-448.3	-340.2	-457.9	-474.5	-219.8	-143.7	-243.2	13.4	

Table[4]

Accuracy: 15/21

Pearl test

818.4	334.2	473.7	676	628	757	837	503	321	322.2	388.7
66.2	734.6	283.8	399.1	491.9	671	675	2952.2	679.1	500.4	

Table[5]

Accuracy: 1/21
Soundgarden test

-40	136.6	640	271.9	313.4	294.3	491.9	529.5	111.5	382	-291
967	312.7	713.7	165.8	335.2	396.7	704.1	328.9	464	418.8	

Table[6]

Accuracy: 20/21
Total accuracy: 36/63 = 57.1%
The success rate is much lower than the first one because these three artists have same genres. It is hard for the classifier to teel.

<u>Test 3</u>
a = 1 situation 1: `pop < threshold1 < rock < threshold2 < classic`
threshold1 =-2.4141e+03
threshold2 =-402.6149
pop < -2414 < rock < -402.6 < classic

pop test

-1920	-2451	-2910	-1609	-2935	-1711	-1930	-1747	-1729	-2567	-3071
-2777	-2039	-1772	-1856	-2304	-2471	-3634	-2560	-2290	-2444	

Table[7]

Accuracy: 9/21
Rock test:

-867.7	-909.7	-1335	-106.1	-1219	-1197	-1060	-1153	-1474	-1573	-1444
-1330	-1193	-1044	-1321	-952	-1032	-1379.	-1218	-1160	-1496	

Table[8]

Accuracy: 20/21
Classic test:

-85.7	-49.3	-568.5	-259.7	-311.7	-479.4	-506.4	-531.9	-268.9	-617.3	-453.8
-365.5	-407.9	-415	-517	-557	-521	-306.3	-885.7	-505	-588.9	

Table[9]

Accuracy: 9/21
Total accuracy: 38/63 = 60.3%
The accuracy is a little bit better than the third one.

Summary

In conclusion, we can build a highly accurate classifier with the combination of LDA and SVD, but only in some kind of linear or simple situation. In test 1, we classify same artists in a high accuracy because the classifier is built based on their own songs. The second test and third test are a little bit hard for LDA and SVD to build an accurate classifier. The problem is that the sample is not enough and same genres may still vary. Moreover, the sampling frequency also matters to LDA. I only choose 21 clips in this project. We can choose more clips in order to build a more accurate classifier.

Citation

[1] "Linear discriminant analysis," *Wikipedia*, 23-Feb-2020. [Online]. Available: https://en.wikipedia.org/wiki/Linear_discriminant_analysis. [Accessed: 07-Mar-2020].
[2] "Makers of MATLAB and Simulink," MathWorks. [Online]. Available: https://www.mathworks.com/. [Accessed: 25-Jan-2020].

Appendix A

[U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A[2]
[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.[2]
[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.[2]
B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B). [2]
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.[2]
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.[2]
s = num2str(A) converts a numeric array into a character array that represents the numbers. [2]
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]
B = sort(A) sorts the elements of A in ascending order.[2]
M = mean(A) returns the mean of the elements of A along the first array dimension whose size does not equal 1.[2]

Appendix B

```
clear; close all; clc

%% test 1
[ariana0] = loading("ariana");
[pink0] = loading("pink");
[chopin0] = loading("chopin");
ariana0 = ariana0';
pink0 = pink0';
chopin0 = chopin0';

[U,S,V,threshold1,threshold2,w,sortpop,sortrock,sortclassic,a] =
dc_trainer(ariana0,pink0,chopin0,20);

a
threshold1
threshold2

% classify
[pval] = classify("pink4.mp3",U,w)

[pval] = classify("ariana3.mp3",U,w)

[pval] = classify("chopin4.mp3",U,w)

%% test 2
[alice0] = loading("alice");
[pearl0] = loading("pearl");
[soundgarden0] = loading("soundgarden");
alice0 = alice0';
pearl0 = pearl0';
soundgarden0 = soundgarden0';

[U,S,V,threshold1,threshold2,w,sortpop,sortrock,sortclassic,a] =
dc_trainer(alice0,pearl0,soundgarden0,20);
a
threshold1
threshold2

% classify
[pval] = classify("alice3.mp3",U,w)

[pval] = classify("pearl3.mp3",U,w)

[pval] = classify("soundgarden3.mp3",U,w)
%% test 3

[pop0] = loading("pop");
[rock0] = loading("rock");
[classic0] = loading("classic");
pop0 = pop0';
rock0 = rock0';
classic0 = classic0';
```

```
[U,S,V,threshold1,threshold2,w,sortpop,sortrock,sortclassic,a] =
dc_trainer(pop0,rock0,classic0,20);
a
threshold1
threshold2

% classify
[pval] = classify("pop3.mp3",U,w)

[pval] = classify("rock3.mp3",U,w)

[pval] = classify("classic4.mp3",U,w)
%% function

function [pval] = classify(songname,U,w)
    song0= [];
    [song,Fs] = audioread(songname);
    song = song'/2;
    song = song(1,:) + song(2,:);
    for j = 20 : 5 : 120
        song1 = song(Fs*j :Fs*(j+5));
        song1_spec = abs(spectrogram(song1));
        song1_spec = reshape(song1_spec, [1,32769*8]);
        song0 = [song0; song1_spec];
    end
    TestMat = U'*song0';  % PCA projection
    pval = w'*TestMat; % LDA projection
end

function [song0] = loading(name)
    song0 = [];
    for i = 1 : 2
        [song,Fs] = audioread(strcat(name,num2str(i),".mp3"));
        song = song'/2;
        song = song(1,:) + song(2,:);
        for j = 20 : 5 : 120
            song1 = song(Fs*j :Fs*(j+5));
            song1_spec = abs(spectrogram(song1));
            song1_spec = reshape(song1_spec, [1,32769*8]);
            song0 = [song0; song1_spec];
        end
    end
end

function [U,S,V,threshold1, threshold2,w,sortpop,sortrock,sortclassic,a] =
dc_trainer(pop0,rock0,classic0,feature)
    np = size(pop0,2); nr = size(rock0,2); nc = size(classic0,2);

    [U,S,V] = svd([pop0 rock0 classic0],'econ');

    SV = S*V'; % projection onto principal components
    U = U(:,1:feature);
    pops = SV(1:feature,1:np);
    rocks = SV(1:feature,np+1:np+nr);
    classics = SV(1:feature, np+nr+1:np+nr+nc);

    mp = mean(pops,2);
    mr = mean(rocks,2);
    mc = mean(classics,2);
    m = (mp+mr+mc)/3;

    Sw = 0; % within class variances
    for k=1:np
        Sw = Sw + (pops(:,k)-mp)*(pops(:,k)-mp)';
    end
    for k=1:nr
        Sw = Sw + (rocks(:,k)-mr)*(rocks(:,k)-mr)';
    end
    for k=1:nc
        Sw = Sw + (classics(:,k)-mc)*(classics(:,k)-mc)';
    end

    Sb = ((mp-m)*(mp-m)' + (mr-m)*(mr-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);

    vpop = w'*pops;
    vrock = w'*rocks;
    vclassic = w'*classics;
    sortpop = sort(vpop);
    sortrock = sort(vrock);
    sortclassic = sort(vclassic);

    if mean(vpop) <= mean(vrock)
        if mean(vrock) <= mean(vclassic)     % pop < threshold1 < rock <
threshold2 < classic
            t1 = length(sortpop);
            t2 = 1;
            while sortpop(t1)>sortrock(t2)
                t1 = t1-1;
                t2 = t2+1;
            end
            threshold1 = (sortpop(t1)+sortrock(t2))/2;
            t3 = length(sortrock);
            t4 = 1;
            while sortrock(t3)>sortclassic(t4)
                t3 = t3-1;
                t4 = t4+1;
            end
            threshold2 = (sortrock(t3)+sortclassic(t4))/2;
            a = 1;
        else
            if mean(vclassic) > mean(vpop)  % pop < threshold1 < classic <
threshold2 < rock
                t1 = length(sortpop);
                t2 = 1;
                while sortpop(t1)>sortclassic(t2)
```

```
                    t1 = t1-1;
                    t2 = t2+1;
                end
                threshold1 = (sortpop(t1)+sortclassic(t2))/2;
                t3 = length(sortclassic);
                t4 = 1;
                while sortclassic(t3)>sortrock(t4)
                    t3 = t3-1;
                    t4 = t4+1;
                end
                threshold2 = (sortclassic(t3)+sortrock(t4))/2;
                a = 2;
            else    % classic < threshold1 < pop < threshold2 < rock
                t1 = length(sortclassic);
                t2 = 1;
                while sortclassic(t1)>sortpop(t2)
                    t1 = t1-1;
                    t2 = t2+1;
                end
                threshold1 = (sortclassic(t1)+sortpop(t2))/2;
                t3 = length(sortpop);
                t4 = 1;
                while sortpop(t3)>sortrock(t4)
                    t3 = t3-1;
                    t4 = t4+1;
                end
                threshold2 = (sortpop(t3)+sortrock(t4))/2;
                a = 3;
            end
        end
    else
        if mean(vpop) <= mean(vclassic)     % rock < threshold1 < pop <
threshold2 < classic
            t1 = length(sortrock);
            t2 = 1;
            while sortrock(t1)>sortpop(t2)
                t1 = t1-1;
                t2 = t2+1;
            end
            threshold1 = (sortrock(t1)+sortpop(t2))/2;
            t3 = length(sortpop);
            t4 = 1;
            while sortpop(t3)>sortclassic(t4)
                t3 = t3-1;
                t4 = t4+1;
            end
            threshold2 = (sortpop(t3)+sortclassic(t4))/2;
            a = 4;
        else
            if mean(vclassic) > mean(vrock)   % rock < threshold1 < classic <
threshold2 < pop
                t1 = length(sortrock);
                t2 = 1;
                while sortrock(t1)>sortclassic(t2)
                    t1 = t1-1;
                    t2 = t2+1;
                end
                threshold1 = (sortrock(t1)+sortclassic(t2))/2;
                t3 = length(sortclassic);
                t4 = 1;
                while sortclassic(t3)>sortpop(t4)
                    t3 = t3-1;
                    t4 = t4+1;
                end
                threshold2 = (sortclassic(t3)+sortpop(t4))/2;
                a = 5
            else    % classic < threshold1 < rock < threshold2 < pop
                t1 = length(sortclassic);
                t2 = 1;
                while sortclassic(t1)>sortrock(t2)
                    t1 = t1-1;
                    t2 = t2+1;
                end
                threshold1 = (sortclassic(t1)+sortrock(t2))/2;
                t3 = length(sortrock);
                t4 = 1;
                while sortrock(t3)>sortpop(t4)
                    t3 = t3-1;
                    t4 = t4+1;
                end
                threshold2 = (sortrock(t3)+sortpop(t4))/2;
                a = 6
            end
        end
    end
end
```