Homework 4: Music Classification

Jiayi Wang

Abstract

In this report, we will see how Singular Value Decomposition (SVD) and Linear Discrimination Analysis (LDA) are being used together to train the computer to help us classify the different bands from different genres, different bands from same genre, and different genres.

Introduction and Overview

Music genres are instantly recognizable to us, whether it is jazz, classical, blues, rap, rock, etc. One can always ask how the brain classifies such information and how it makes a decision based upon hearing a new piece of music. The objective of this problem set is to attempt to write a code that can classify a given piece of music by sampling a 5 second clip. We will first obtain the music legally on the source website, such as: YouTube audio library, Sound cloud and Free Music Archive, which were provided and chop 5 second clip from each music piece we obtain. We will SVD the spectrogram of music, then use statistical information about the SVD decomposition and LDA together in order to classify the different bands from different genres, different bands from same genre, and different genres. At last, we will test the accuracy of our training algorithm on our test data to see how well the computer can classify the music pieces.

Theoretical Background

- **Singular Value Decomposition (SVD)**

  A second method for diagonalizing the covariance matrix is the SVD method. In this case, the SVD can diagonalize any matrix by working in the appropriate pair of bases U and V as outlined in the first lecture of this section. Thus, by defining the transformed variable:

$$\mathbf{Y} = \mathbf{U}^{*}\mathbf{X} \tag{1}$$

where U is the unitary transformation associated with the SVD: $X = U\Sigma V^*$. Just as in the eigenvalue/eigenvector formulation, we then compute the variance in Y:[1]

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{\Sigma}^2 \tag{2}$$

- **Linear Discrimination Analysis (LDA)**

The idea of the LDA was first proposed by Fisher in the context of taxonomy. In our example, two data sets are considered and projected onto new bases. In the left figure, the projection shows the data to be completely mixed, not allowing for any reasonable way to separate the data from each other. In the right figure, which is the ideal caricature for LDA, the data are well separated with the means $\mu 1$ and $\mu 2$ being well apart when projected onto the chosen subspace. Thus, 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. Construct a projection w such that:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \tag{3}$$

where the scatter matrices for between-class S_B and within-class S_W data are given by[2]:

$$\mathbf{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{4}$$

$$\mathbf{S}_W = \sum_{j=1}^{2} \sum_{\mathbf{x}} (\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^T \tag{5}$$

For a n-class LDA, S_B and S_W are given by:

$$S_B = \sum_{1}^{n} m_j (\mu_j - \mu)(\mu_j - \mu)^T \tag{6}$$

$$S_W = \sum_{j=1}^{n} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{7}$$

where m_j is the number of samples from class j.

[1] Page 406 from course notes.
[2] Page 457 from course notes.

Algorithm Implementation and Development

To prepare the music pieces, we will download the music in mp3 from the websites: YouTube audio library, Sound cloud and Free Music Archive which were provided in the instruction. Then with the help of the website: *https://audio.online-convert.com*, we are able to change the audio channels to mono, chop a five second clip from each mp3 file (since each music piece are more than one minute thirty seconds, so I choose this five second clip to be from 00:01:00 to 00:01:05), and convert them to wav file.

To load the five second music clips into Matlab, we use the command called *dir()* to lists all files and folders in the folder where five second music clips locate and save them in a variable called *file*. Then by using *file.name*, we will be able to obtain all the file names of the music pieces, and we will save them in a variable called *songs*. We will get rid of the first two elements in the array *songs* since there are empty and restore the rest of the names in *songs*.

Then we build a for loop. Inside of the for loop, we first will get the names of the songs and use the command *[y,Fs]= audioread()* to obtain sampled data, *y*, and a sample rate for that data, *Fs*. Next, we resample the data by using the command *resample()* in order to take every fourth point for more manageable data sizes. Then we concatenate these data into a matrix which was initiated to represent that kind of music. In this matrix, each column represents one music piece. In each test, we will use three of these for loops to obtain three matrices which represent three different kind of music that need to be classified according to the case requirement.

Next, we obtain the spectrogram of each music piece and get rid of the imaginary part by using *abs(spectrogram())*, reshape them back into the column vector by using *reshape()* and concatenate all these columns from three kinds of music into a big matrix. Then we SVD the spectrogram of songs by using *[U, S, V]=svd(... , 'econ')*. Then we obtain the projection onto principal components using *S*V'*, and we will save that into a variable called *music*. For the best accuracy, we will find one feature number that will give us the best result (different in each test) and we will only take the first feature number columns of *U*.

Now we can calculate the between-class S_B and within-class S_W in formula (6) and (7) by giving the matrix *music*. Then LDA can be applied by first using *[V2,D]=eig()* command, since in order to get the LDA projection basis *w*, we will first find the maxima eigenvalue in absolute value, which locates in the diagonal entries of *D*. Then we can easily find this *w* in *V2* at its corresponding location. Finally, we can project the music pieces from three groups onto this vector *w* to get points to represent these music pieces.

To classify the music belongs to which of the three group. We need to find two thresholds. First, we will sort the mean of the points from the same group, we will call them group one, group two and group three with increasing mean value. To find the threshold between group one and group two, we will sort the numbers in each group, and compare the first number of group one with the last number in group two. While the first number of group one is smaller than the last number in group two, we will increase the index from group one and decrease the index from group two one at a time until we find the value in group one that is greater than the value in group two. Then we set the value of threshold one to be the average value of these two number. We will use the same strategy to find threshold two.

After we have the training algorithm done, we are able to test the accuracy. The test set will undergo the same process as the training music pieces. At last, with the help of two thresholds, we are able to classify the music pieces from different groups and calculate the accuracy of it.

Computational Results

In test 1, the training data are from three different Chinese singers and of different genres. These three singers are Ziqi Deng, Jielun Zhou, and Jingru Liang. In test 2, the training data are from three different bands but within the same genre. These three bands are: Soundgarden, Alice in Chains, and Pearl Jam. In test 3, the training data are from three different genres. These three genres are: country, jazz, and hiphop. In each test, training data set is consisting of 60 music pieces with 20 pieces from each group; and the test set is consisting of 15 music pieces.

From table 1 below, we can see that the accuracy in correctly classifying a 5-second sound clip from three different bands but within the same genre (test 2) are much lower than the accuracy in correctly classifying a 5-second sound clip from three different singers of different genres (test 1). This matches what we expected, since in test 2 the songs from all three bands are within same genre; they share many of the same characteristics, which makes classification way more difficult. So lower accuracy is expected.

In test three, since genres are very different, the classifier did a reasonable job on telling the difference between music pieces. This matches our expectation.

Another thing we notice is the accuracies are not as high as we would expect. We believe it is due to the small training set size. If we increase the training set size, higher accuracy should be expected. At the same time, if we increase the size of testing set, the accuracy of the success rate will also be more persuasive.

Test Number	Accuracy
1	53.33%
2	33.33%
3	66.67%

Table 1: Classification accuracy in all three tests

Summary and Conclusions

By applying SVD and LDA, we are able to separate two or more classes of objects or events. As we can see in our tests, the performance of our classifier increases when there are big differences between training data groups. The performance decreases when there is complexity added in each group or more similarities between the groups. In order to increase the accuracy, larger training set size should be introduced. At the same time, if we increase the size of testing set, the accuracy of the success rate will also be more persuasive.

Appendix A. MATLAB functions used and brief implementation explanation

- *Listing=dir(name)* lists the contents of the folder. Specify name using absolute or relative path names.
- *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.
- *[y,Fs] = audioread(filename)* reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
- *y = 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.
- *[U,S,V] = svd(A,'econ')* produces an economy-size decomposition of m-by-n matrix A.
- *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. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).[3]

Appendix B. MATLAB codes

```
%% Test case 1
clear all;close all;clc
path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\Deng';
file=dir(path);
songs={file.name};
songs=songs(3:end);
dengdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%     sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%    sound(y,Fs/4)
    dengdata=[dengdata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\Liang';
file=dir(path);
```

[3] All the MATLAB functions and their brief implementation explanation are cited from www.mathwork.com.

```
songs={file.name};
songs=songs(3:end);
liangdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    liangdata=[liangdata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\Zhou';
file=dir(path);
songs={file.name};
songs=songs(3:end);
zhoudata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    zhoudata=[zhoudata y];
end
deng_spec=spect(dengdata);
liang_spec=spect(liangdata);
zhou_spec=spect(zhoudata);
[U,S,V,threshold1,threshold2,w,m1,m2,m3]=music_trainer(deng_spec,liang_spec,z
hou_spec,56);

%% Test case 2
clear all;close all;clc

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\AliceInChains';
file=dir(path);
songs={file.name};
songs=songs(3:end);
AliceInChainsdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    AliceInChainsdata=[AliceInChainsdata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\PearlJam';
file=dir(path);
songs={file.name};
songs=songs(3:end);
PearlJamdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
```

```
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    PearlJamdata=[PearlJamdata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\Soundgarden';
file=dir(path);
songs={file.name};
songs=songs(3:end);
Soundgardendata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    Soundgardendata=[Soundgardendata y];
end
AliceInChains_spec=spect(AliceInChainsdata);
PearlJam_spec=spect(PearlJamdata);
Soundgarden_spec=spect(Soundgardendata);
[U,S,V,threshold1,threshold2,w,m1,m2,m3]=music_trainer(AliceInChains_spec,Pea
rlJam_spec,Soundgarden_spec,30);


%% Test case 3
clear all;close all;clc

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\country';
file=dir(path);
songs={file.name};
songs=songs(3:end);
countrydata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    countrydata=[countrydata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\hiphop';
file=dir(path);
songs={file.name};
songs=songs(3:end);
hiphopdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    hiphopdata=[hiphopdata y];
end

path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\jazz';
```

```
file=dir(path);
songs={file.name};
songs=songs(3:end);
jazzdata=[];
for i=1:20
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    jazzdata=[jazzdata y];
end
country_spec=spect(countrydata);
hiphop_spec=spect(hiphopdata);
jazz_spec=spect(jazzdata);
[U,S,V,threshold1,threshold2,w,m1,m2,m3]=music_trainer(country_spec,hiphop_sp
ec,jazz_spec,25);

%% Test classifier
clc
%path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\test1';
path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\test2';
%path='C:\Users\jacke\OneDrive\Documents\MATLAB\AMATH482HW4\test3';
file=dir(path);
songs={file.name};
songs=songs(3:end);
testdata=[];
for i=1:15
    filename=strcat(path,'\',char(songs(i)));
    [y,Fs]=audioread(filename);
%    sound(y,Fs)
    y=resample(y,Fs/4,Fs);
%   sound(y,Fs/4)
    testdata=[testdata y];
end

test_spec=spect(testdata);
testMat=U'*test_spec;
pval=w'*testMat;
testresult=[];
for i=1:size(pval,2)
    if pval(i)<threshold1
        testresult=[testresult 1];
    elseif pval(i)>threshold1 && pval(i)<threshold2
        testresult=[testresult 2];
    else
        testresult=[testresult 3] ;
    end
end
%answer=[1 1 1 1 1 3 3 3 3 3 2 2 2 2 2];   %for test 1
answer=[1 2 2 2 1 2 3 3 3 3 1 3 1 1 2];  %for test 2
%answer=[2 3 2 3 1 1 3 3 2 2 1 3 1 1 2];  %for test 3
yes=0;
for i=1:15
    if answer(i)==testresult(i);
        yes=yes+1;
    end
```

```
end
disp('Rate of success');
sucRate = yes/15

% trainer function
function
[U,S,V,threshold1,threshold2,w,m1,m2,m3]=music_trainer(country0,hiphop0,jazz0
,feature)
    nc=size(country0,2);
    nh=size(hiphop0,2);
    nj=size(jazz0,2);
    [U,S,V]=svd([country0 hiphop0 jazz0],'econ');
    %plot(diag(S)/sum(diag(S)), 'ro', 'Linewidth', 2)
    music=S*V'; %projection onto principal components
    U=U(:,1:feature);
    country=music(1:feature,1:nc);
    hiphop=music(1:feature,nc+1:nc+nh);
    jazz=music(1:feature,nc+nh+1:nc+nh+nj);
    mc=mean(country,2);
    mh=mean(hiphop,2);
    mj=mean(jazz,2);
    mtotal=mean(music(1:feature,:),2);

    Sw=0;
    for k=1:nc
        Sw=Sw+(country(:,k)-mc)*(country(:,k)-mc)';
    end
    for k=1:nh
        Sw=Sw+(hiphop(:,k)-mh)*(hiphop(:,k)-mh)';
    end
    for k=1:nj
        Sw=Sw+(jazz(:,k)-mj)*(jazz(:,k)-mj)';
    end

    Sb=nc.*(mc-mtotal)*(mc-mtotal)'+nh.*(mh-mtotal)*(mh-mtotal)'+nj.*(mj-
mtotal)*(mj-mtotal)';

    [V2,D]=eig(Sb,Sw);
    [~,ind]=max(abs(diag(D)));
    w=V2(:,ind); w=w/norm(w,2);
    vcountry=w'*country;
    vhiphop=w'*hiphop;
    vjazz=w'*jazz;

    m1=mean(vcountry);
    m2=mean(vhiphop);
    m3=mean(vjazz);
    if m1==max([m1 m2 m3])
        if m2==min([m1 m2 m3])
            [threshold1,threshold2]=thresh(vhiphop,vjazz,vcountry);
        else
            [threshold1,threshold2]=thresh(vjazz,vhiphop,vcountry);
        end
    elseif m2==max([m1 m2 m3])
        if m1==min([m1 m2 m3])
            [threshold1,threshold2]=thresh(vcountry,vjazz,vhiphop);
        else
```

```
            [threshold1,threshold2]=thresh(vjazz,vcountry,vhiphop);
        end
    else
        if m1==min([m1 m2 m3])
            [threshold1,threshold2]=thresh(vcountry,vhiphop,vjazz);
        else
            [threshold1,threshold2]=thresh(vhiphop,vcountry,vjazz);
        end
    end
end

% find two thresholds
function [threshold1,threshold2]=thresh(mi,me,ma)

    sortmi=sort(mi);
    sortme=sort(me);
    t1 = length(sortmi);
    t2 = 1;
    while sortmi(t1)>sortme(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sortmi(t1)+sortme(t2))/2;

    sortme=sort(me);
    sortma=sort(ma);
    t1 = length(sortme);
    t2 = 1;
    while sortme(t1)>sortma(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sortme(t1)+sortma(t2))/2;
end

% spectrogram function
function spe=spect(data)
    spe=[];
    for i=1:size(data,2)
        spe(:,i)=reshape(abs(spectrogram(data(:,i))),[],1);
    end
end
```