Homework 4: Music Classification

Minghu Zhou

February 29, 2020

Abstract

Music genres are instantly recognizable to us, whether it be 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. In this report, we use Linear Discriminant Analysis (LDA) to classify a given piece of music by sampling a 5 second clip.

1. Introduction and Overview

The basic idea is that we give our machine some data of different groups, the machine can learn the pattern of each group. When we give the machine new data, it will be able to classify them.

There are three cases within which we train and classify music clips. In the first case, there are three different singers and their music genres are different. In the second case, we have three different singers but their music genres are the same. In the third case, we train three genres, each including music clips of different singers and then test if the machine knows which genre a new music clip belongs to. We can see the accuracy of each case varies a lot.

2. Theoretical Background

2.1 Singular Value Decomposition(SVD)

The SVD allows a matrix to be reformatted. A matrix X can be reformatted into the following:

$$X = U\Sigma V^T \tag{2.1}$$

where U and V are unitary and Σ is diagonal. With unitary transformation matrix U as a basis, the transformed variable Y can be defined as:

$$Y = U^{-1}X = U^T X \tag{2.2}$$

2.2 Principal component Analysis (PCA)

Principal Component Analysis is used with SVD to reduce the dimensionality of the data. It allows us to get the low dimensional projection of the data. We can use the first few principal components to capture the major pattern of data. In particular, we just need some major features of the songs clips we trained. To achieve this, we take the first few columns of U after the SVD of the data matrix X. In this way, we reduce the dimensionality of the transformed data.

2.3 Linear Discriminant Analysis (LDA)

Say we have several sample data sets. Using LDA we can project them onto a new basis. On this basis, the variance within class is minimized and the variance between class is maximized.

Within-class scatter matrix is defined as:

$$S_w = \sum_{j=1}^{n} \sum_{\vec{x}} (\vec{x} - \overrightarrow{\mu_J})(\vec{x} - \overrightarrow{\mu_J})^T \qquad (2.3)$$

Where $\vec{x}$ is each data set and $\overrightarrow{\mu_J}$ is the mean of each data set.

Between-class scatter matrix is defined as:

$$S_B = \sum_{j=1}^{n} m_j(\overrightarrow{\mu_J} - \vec{\mu})(\overrightarrow{\mu_J} - \vec{\mu})^T \qquad (2.4)$$

Where m_j is the number of samples in each data set, $\overrightarrow{\mu_J}$ is the mean of each data set and $\vec{\mu}$ is the mean of all data sets.

The projection w is defined as:

$$\vec{w} = \arg\max \left(\frac{\overrightarrow{w^T} S_B \vec{w}}{\overrightarrow{w^T} S_W \vec{w}}\right) \qquad (2.5)$$

We form the generalized eigenvalue problem to find the solution:

$$S_B \vec{w} = \lambda S_W \vec{w} \qquad (2.6)$$

We are interested in the eigenvector related to the largest eigenvalue. This eigenvector is our projection basis.

3. Algorithms Implementation and Development

3.1 Data collection

The Algorithm is almost the same for three cases. First, we load in songs of different genres we collect online. Note that the songs are stereo, which means they are played in two channels so there are two columns of data in the matrix loaded in. To make each song mono, we average the two columns and get a new vector. We want to re-sample our data to keep the data sizes more manageable, so we pick every fourth point. Meanwhile, we need to divide *Fs* which is samples per second by a factor of 4. We cut the beginning and ending zeros in the vector so we get rid of the soundless part. Then we reshape the vector to a matrix where each column corresponds to a 5-second clip.

3.2 Training

Iterate over songs within a genre, we combine the resulting matrix and apply Fast Fourier Transform to it to transform our data to their frequency space. Now we have three matrices corresponding to three genres, we combine them to get a larger matrix. We apply *svd* to it and get the corresponding U, S, V. Using $S * V'$ we project the data onto principal components. Since we only need to capture a few features, we take the first few columns of U. Then we separate out each group from the projected matrix, each matrix has size of feature numbers times sample sizes within each group. Next, the Within-class scatter matrix and Between-class scatter matrix can be calculated using the mean of each matrix across the row and that of the combined lager matrix. The *eig* command allows us to find the eigenvalues. We can also find eigenvector w related to the largest eigenvalue. Then we project each group to the w basis so each sample is now a one-dimension data.

Now we have three groups of one-dimension data. We sort them and determine their order by their means. Then we need to find thresholds between two neighboring group. We start from the last element a of the group with lower mean and the first element b of the group with the larger mean. If a is greater than b, we go forward by one index in the group with the lower mean and go backward by one index in the group with the larger mean. We repeat this process until a is smaller than b. Average a and b we will get the threshold value between these two groups. In this way, we make sure that the number of outliers is almost the same for each group using the resulting threshold value. Using this method, we can find two threshold values for three groups.

3.3 Testing

Now we start to test new data. We load in songs of different genres and repeat the process of 3.1. We will get a large matrix where each column represents the spectrum of each sample. We project the matrix onto principal components by multiplying U^T, and then project onto w basis. The resulting vector is the one-dimension value of each test clip. Compare each element in this vector to the two thresholds we create a vector called *testLabel* which records the group of each test clip classified by our algorithm. We also build a vector called *hiddenLabel* where each element (either 1, 2 or 3) corresponds to the actual group in that index. Subtracting one vector by another and getting the number of non-zero elements we will get the number of wrong classification. Using 1 minus this number divided by the number of test clips we can get the accuracy on the test set.

Repeat all of the above process for three cases we can get the accuracy for each case.

4. Computational Results

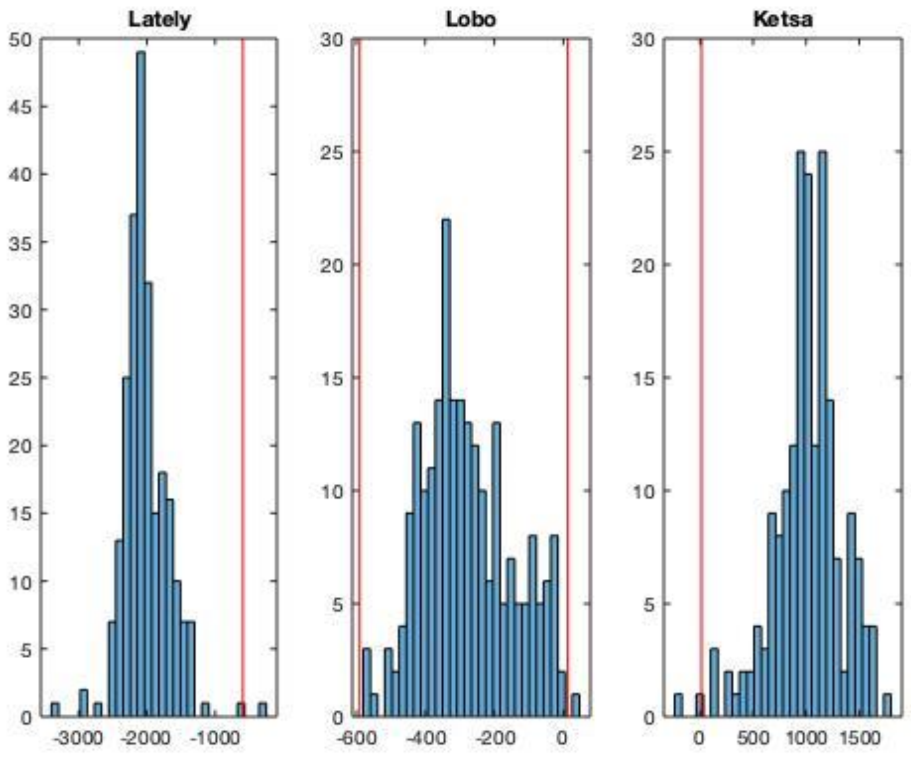

Figure 1: projected values distribution for Case 1: three different bands of different genres

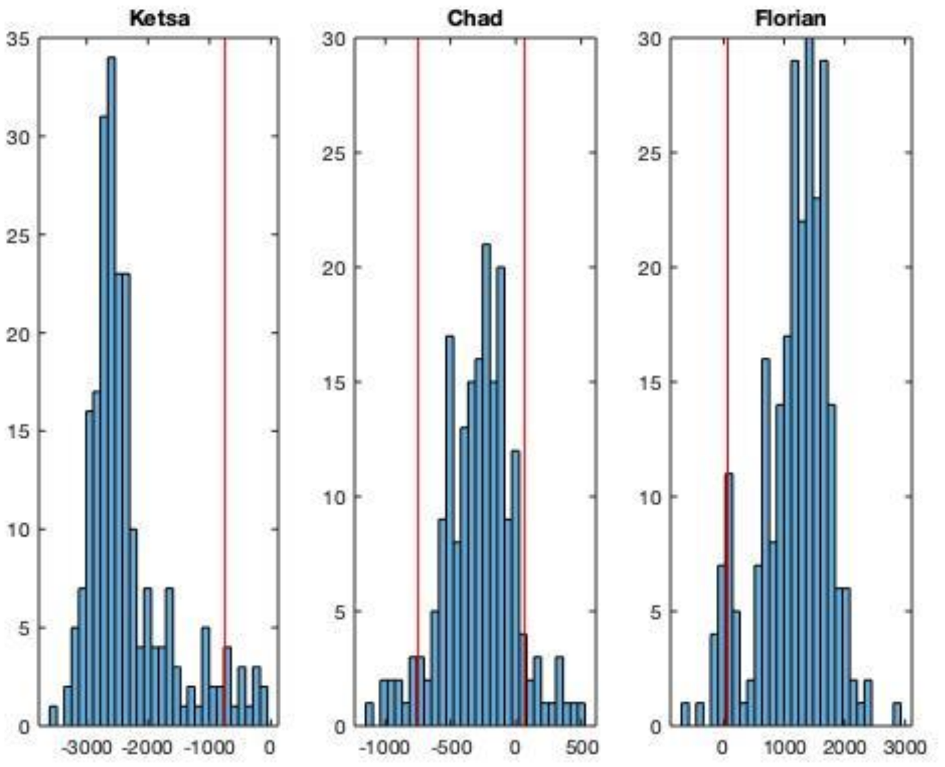

Figure 2: projected values distribution for Case 2: three different bands within the same genre

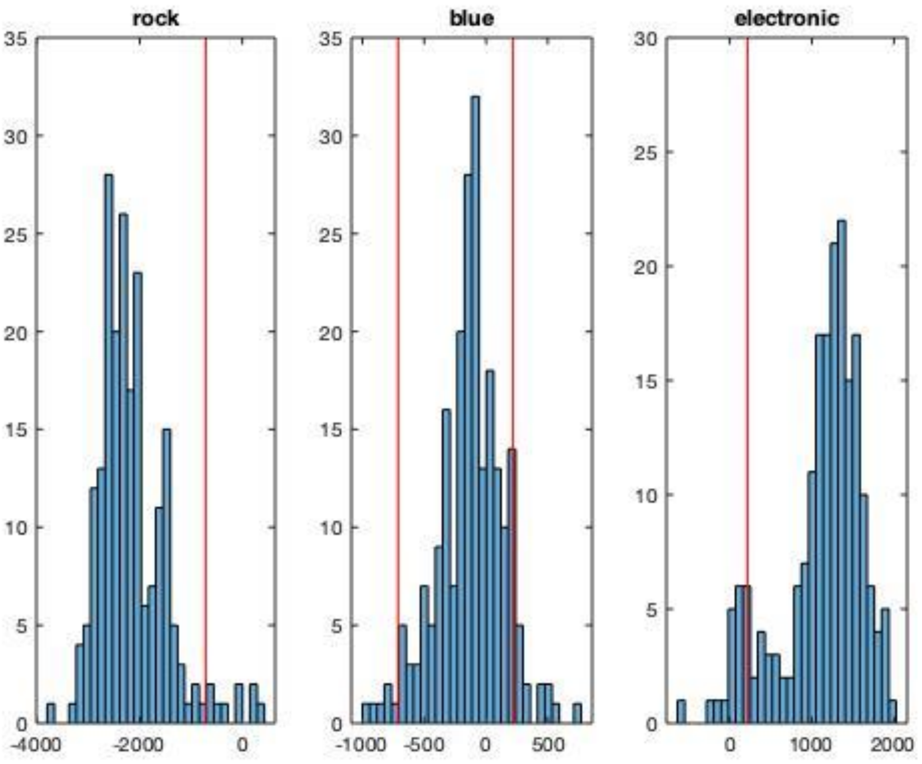

Figure 3: projected values distribution for Case 3: various bands within each genre

Case 1: three different bands of three different genres

The threshold for this case are -590.11 and 12.79. The accuracy on the test set is 91.11%. From Figure 1 we see the number of outliers in each group is very small, which means neighboring groups barely overlap with eac other. This is due to the three bands are in different genres so their styles are very different. As a result, new clips of these bands are easy to classify and we see the accuracy is very high.

Case 2: three different bands within the same genre

The threshold for this case are -747.48 and 65.40. The accuracy on the test set is 63.43%. From Figure 2 we see the number of outliers in each group is much larger than that in Figure 1, which means neighboring groups overlap with each other to some extent. This is because the bands are within the same genre so their styles are somewhat same. Consequently, new clips of these bands are hard to classify and we see the accuracy is much less than that in case 1.

Case 3: various bands within the same genre

The threshold for this case are -710.81 and 215.76. The accuracy on the test set is 72.36%. From Figure 3 we see the number of outliers in each group is still larger than that in Figure 1, especially for the blue and electronic group. Even though we take samples of different genres, these samples are from different bands within each genre. This brings difficulties when classifying test clips of other bands so we see the accuracy is not very high.

5. Summary and Conclusions

Using Linear Discriminant Analysis, we reduce dimensionality of our data and build thresholds between groups. We apply this algorithm to three cases which are three different bands of three different genres, three different bands within the same genre and various bands within the same genre. We see high accuracy in the first case and comparatively low accuracy in the second and third case. This is due to the samples in each group are very similar while each group is very distinctive. Thus, new test set can be easily classified without much error. The genre of each group is the same In the second case while the sample clips in each group are somewhat different in the third case, so the accuracy of classification of test clips is not very high in these two groups.

Appendix A.
MATLAB functions and brief implementation explanation

D = diag(v) returns a square diagonal matrix with the elements of vector v on the main diagonal.

k = find(X) returns a vector containing the linear indices of each nonzero element in array X.

[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'.

B = sort(A) sorts in ascending order. The sorted output B has the same type and size as A.

[V,D] = eig(A) produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = V*D.

reshape(X,[M,N]) returns the M-by-N matrix whose elements are taken columnwise from X. An error results if X does not have M*N elements.

Appendix B.
MATLAB codes

```
%% first case
clear all; close all;clc
Lobo_spec = data("training1/","Lobo",5);
Ketsa_spec = data("training1/","Ketsa",5);
Lately_spec = data("training1/","Lately",5);

[U,S,V,order,threshold,w,sortband1,sortband2,sortband3] =
bands_trainer(Lobo_spec,Ketsa_spec,Lately_spec,40);
plotting(threshold,order,{sortband1,sortband2,sortband3},["Lobo","Ketsa","L
ately"]);
```

```
Lobo_test = data("testing1/","Lobo",1);
Ketsa_test = data("testing1/","Ketsa",1);
Lately_test = data("testing1/","Lately",1);

bands_tester(Lobo_test,Ketsa_test,Lately_test,U,w,threshold,order);
%% second case
clear all; close all;clc
Chad_spec = data("training2/","Chad",6);
Florian_spec = data("training2/","Florian",5);
Ketsa_spec = data("training2/","Ketsa",6);

[U,S,V,order,threshold,w,sortband1,sortband2,sortband3] =
bands_trainer(Chad_spec,Florian_spec,Ketsa_spec,40);
plotting(threshold,order,{sortband1,sortband2,sortband3},["Chad","Florian",
"Ketsa"]);

Chad_test = data("testing2/","Chad",1);
Florian_test = data("testing2/","Florian",1);
Ketsa_test = data("testing2/","Ketsa",1);

bands_tester(Chad_test,Florian_test,Ketsa_test,U,w,threshold,order);
%% third case
clear all; close all;clc
blue_spec = data("training3/","blue",5);
electronic_spec = data("training3/","electronic",5);
rock_spec = data("training3/","rock",5);

[U,S,V,order,threshold,w,sortband1,sortband2,sortband3] =
bands_trainer(blue_spec,electronic_spec,rock_spec,40);
plotting(threshold,order,{sortband1,sortband2,sortband3},["blue","electroni
c","rock"]);

blue_test = data("testing3/","blue",1);
electronic_test = data("testing3/","electronic",1);
rock_test = data("testing3/","rock",1);

bands_tester(blue_test,electronic_test,rock_test,U,w,threshold,order);
%%
function bands_tester(test_spec1,test_spec2,test_spec3,U,w,threshold,order)
    size1 = size(test_spec1,2);
    size2 = size(test_spec2,2);
    size3 = size(test_spec3,2);
    hiddenLabel = zeros(1, size1+size2+size3);
    hiddenLabel(1:size1) = 1;
    hiddenLabel(size1+1:size1+size2) = 2;
    hiddenLabel(size1+size2+1:end) = 3;

    Test_spec = [test_spec1 test_spec2 test_spec3];
    TestMat = U'*Test_spec;  % PCA projection
    pval = w'*TestMat;  % LDA projection
    testLabel = zeros(1,length(pval));
    for i = 1:length(pval)
        if pval(i) <= threshold(1)
            testLabel(i) = order(1);
        elseif pval(i)<=threshold(2)
            testLabel(i) = order(2);
        else
            testLabel(i) = order(3);
        end
    end
    diff = (testLabel - hiddenLabel) ~= 0;
    sucRate = 1 - sum(diff)/length(diff);
    display(sucRate);
end
%%

function plotting(threshold,order,sortbands,titles)
   small = sortbands{1,order(1)};
   small_title = titles(order(1));
   mid = sortbands{1,order(2)};
   mid_title = titles(order(2));
   big = sortbands{1,order(3)};
   big_title = titles(order(3));

   figure()
   subplot(1,3,1)
   histogram(small,30);
   hold on
   plot([threshold(1) threshold(1)],[0 35],'r')
   title(small_title)
   subplot(1,3,2)
   histogram(mid,30);
   hold on
   plot([threshold(1) threshold(1)],[0 35],'r')
   plot([threshold(2) threshold(2)],[0 35],'r')
   title(mid_title)
   subplot(1,3,3)
   histogram(big,30);
   hold on
   plot([threshold(2) threshold(2)],[0 30],'r')
   title(big_title)
end

function [U,S,V,order,threshold,w,sortband1,sortband2,sortband3] =
bands_trainer(b1,b2,b3,feature)
    n1 = size(b1,2);
    n2 = size(b2,2);
    n3 = size(b3,2);

    [U,S,V] = svd([b1 b2 b3],'econ');
    bands = S*V'; % projection onto principal components
    U = U(:,1:feature);

    band1 = bands(1:feature,1:n1);
    band2 = bands(1:feature,n1+1:n1+n2);
    band3 = bands(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(band1,2);
    m2 = mean(band2,2);
    m3 = mean(band3,2);
    m = mean(bands(1:feature,:),2);

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (band1(:,k)-m1)*(band1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (band2(:,k)-m2)*(band2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (band3(:,k)-m3)*(band3(:,k)-m3)';
    end

    Sb = n1*(m1-m)*(m1-m)'+n2*(m2-m)*(m2-m)'+n3*(m3-m)*(m3-m)'; % between
class

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

    vband1 = w'*band1;
    vband2 = w'*band2;
    vband3 = w'*band3;

    sortband1 = sort(vband1);
    sortband2 = sort(vband2);
    sortband3 = sort(vband3);

    sortbands = {sortband1, sortband2, sortband3};
    bands_mean = [mean(sortband1) mean(sortband2) mean(sortband3)];
    bands_mean_sorted = sort(bands_mean);
    order = zeros(1,3);
    for i = 1:3
        order(i) = find(bands_mean == bands_mean_sorted(i));
    end

    sortband_small = sortbands{1,order(1)};
    sortband_mid = sortbands{1,order(2)};
    sortband_big = sortbands{1,order(3)};

    threshold = zeros(1,2);

    t1 = length(sortband_small);
    t2 = 1;
    while sortband_small(t1)>sortband_mid(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold(1) = (sortband_small(t1)+sortband_mid(t2))/2;

    t1 = length(sortband_mid);
    t2 = 1;
    while sortband_mid(t1)>sortband_big(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold(2) = (sortband_mid(t1)+sortband_big(t2))/2;
end

function spec = data(path, band,number_songs)

    spec = [];
     for j = 1:number_songs
         [song, Fs] = audioread(path+band+num2str(j)+".mp3");
         Fs = Fs / 4;

         song = song(1:4:end,:);
         monosong = mean(song,2);
```

```
          monosong =
monosong(find(monosong ,1,'first'):find(monosong,1,'last'));
          %5 second intervals
          pieceLength = Fs*5;
          number = floor(length(monosong) / pieceLength);
          monosong = monosong(1:(pieceLength * number));
          data = reshape(monosong, [pieceLength, number]);

          spec = [spec , data];
     end
     spec = abs(fft(spec));

end
```