Music Genre Classification

Yinxi Pan

AMATH482, Homework4, Winter2020

March 7, 2020

Abstract

The report talks about how to perform music classifications in Matlab. The main method used is the Linear Discriminant Analysis. Perform classification by sampling 5-second clips from a variety of music. The report classifies three different cases. The first one is classifying songs from three different bands in three different genres. The second one is classifying three different bands in the Seattle area. The last one is classifying three different genres in general where the singer is different from each song. The accuracy achieved in the first test is 63.64% The accuracy achieved in the second test is 60.61% The accuracy achieved in the third test is 42.42%.

1 Introduction and Overview

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[1]. In this report, we show how Linear Discriminant Analysis helps us build a music classifier in Matlab by sampling 5-second clips. We build three classifiers in this homework. The first one is to classify different bands in three different genres. The second one is to classify three different bands in the Seattle area. The last one is to classify three different genres in general. For the first test, I choose three singers: Jackson Yee, EXO, and Beethoven. Jackson represents the c-pop music; EXO represents the k-pop music and Beethoven represent classical music. Ideally, we would expect a high accuracy with the classifier since they are quite different from each other. For the second test, as suggested in the spec, I choose Soundgarden, Alice in Chains, and Pearl Jam to test the classifier for the Seattle area. For the last test, I choose the bank rock music, classical music, and country music, with singers, suggested in the spec.

[1]From homework spec.

2 Theoretical Background

Linear Discrimination Analysis (LDA)[2]

Linear discriminant analysis[3] (LDA)[4] is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification. LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA, in contrast, does not take into account any difference in class, and factor analysis builds the feature combinations based on differences rather than similarities. Discriminant analysis is also different from factor analysis in that it is not an interdependence technique: a distinction between independent variables and dependent variables (also called criterion variables) must be made. So the overall goal of LDA is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the inter-class data. Equation(1) is the projection w under a 2-class(S_W, S_B) LDA

$$w = \text{argmax}\frac{w^T S_B w}{w^T S_w w} \tag{1}$$

where the scatter matrices for between-class S_B and within-class S_W data are given by Equation(2) and Equation(3)

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{2}$$

$$\sum_{j=1}^{2}\sum_{x}^{\infty}(x - \mu_j)(x - \mu_j)^T \tag{3}$$

These quantities essentially measure the variance of the data sets as well as the variance of the difference in the means. The criterion given by Equation(1) is commonly known as the generalized Rayleigh quotient whose solution can be found via the generalized eigenvalue problem as in equation(4)

$$S_B w = \lambda S_W w \tag{4}$$

where the maximum eigenvalue and its associated eigenvector gives the quantity of interest and the projection basis.

[2]Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

[3]Singular value decomposition and Principal Component Analysis have been introduced in the last report.

[4]https://en.wikipedia.org/wiki/Linear$_{d}iscriminant_{a}nalysis$

3 Algorithm Implementation and Development

The implementation are similar among all cases. Below is the illustration of the algorithm for the first case. For different cases, we only need to change the threshold.[5] To change the threshold, we need to plot the trained dataset first and then observe the order of threshold. The algorithm can be briefly summarized as

1. Project the images onto the SVD modes selected from the training algorithm.
2. Project this three-level decomposition onto the LDA eigenvector.
3. Determine the projection value relative to the LDA threshold determined in the training algorithm.
4. Classify type of music.

To be more specific:

1. Load the music files and build the training dataset. We create a loop to go over all the music files, covert them into spectrogram using Matlab built-in function and reshape them into the data matrix. Note all music files have 2 columns of sample data. This is because they are classical and we could either take one or add them up. Here I only trained the classifier with 2 songs, each 20 clips, from each category.
2. Perform singular value decomposition and find the threshold between each category. To figure out the order of the threshold, we plot the sorted projection(Figure(1,3,5) to determine the order of the threshold. Then we build the thresholds by comparing the magnitude of projections. Here to build the classifier, we need to take the mean of all there class rather than shown in the previous section.
3. Build the testing dataset. The way to build the testing dataset is the same as the training dataset. But we choose a different starting point and different number of clips. Here I choose 10 clips which is one half number of the clips used to train. Then we use the U matrix from SVD performed before and create the LDA basis and projection.
4. Test the accuracy. Create a matrix that represent the true value(underlying label) and subtract from the LDA projection matrix. Calculate the number of 0 in the resulting matrix which tells the accuracy of our classifier.

4 Computational Results

1. (test 1) Band Classification. Accuracy = 63.64%

   Figure(1) is the plot of the training data projection on w. As we could see, there a prominent difference between each category. However, in Figure(2), after we test the classifier

[5]Matlab code please check Appendix.B

with different data, the threshold doesn't work pretty good since we would expect the left is below the horizontal lines and the right should be above the horizontal line. This might relate to the place we take the clips. Also, it could be that these songs have some underlying musical relationship.

2. (test 2) The Case for Seattle. Accuracy = 60.61%

   Figure(3) is the plot of the training data projection on w. As we could see, there a prominent difference between each category. However, in Figure(4), after we test on classifier with different data, the threshold doesn't work pretty good since we would expect the left is below the horizontal lines and the right should be above the horizontal line. I suspect the same thing as test2.

3. (test 3) Genre Classification. Accuracy = 42.42%

   Figure(5) is the plot of the training data projection on w. As we could see, there a prominent difference between each category. However, again, in Figure(6), the threshold doesn't work pretty good with the same suspected reason.

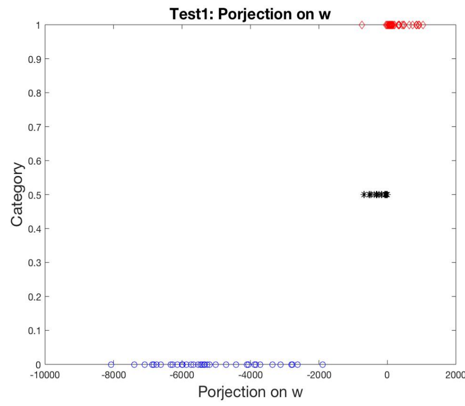

Figure 1: Test1: Projection on w for different categories. The red one is EXO; the black one is Beethoven; the blue one is Jackson.

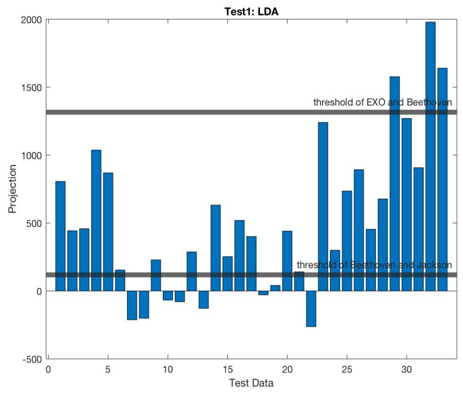

Figure 2: Test1: Bar plot of the statistics associated with the three categorical music data once projected onto the LDA basis.

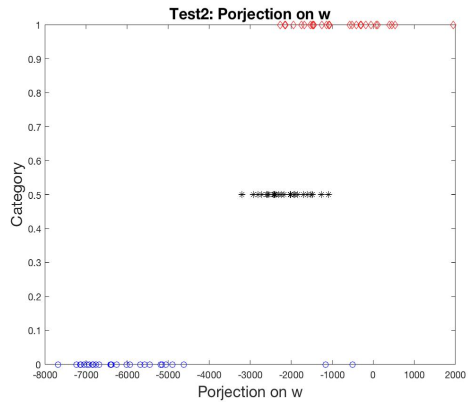

Figure 3: Test2: Projection on w for different categories. The red one is Alice; the black one is Pearl; the blue one is Soundgarden.

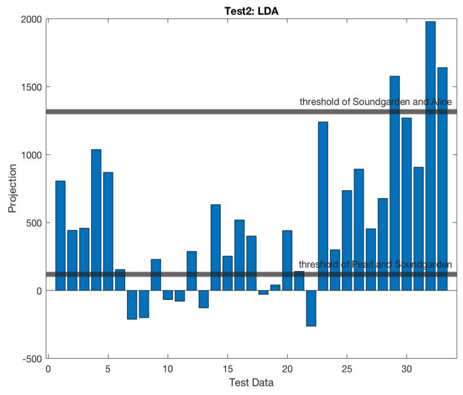

Figure 4: Test2: Bar plot of the statistics associated with the three categorical music data once projected onto the LDA basis.

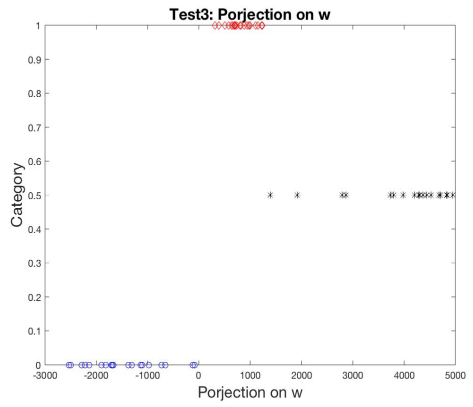

Figure 5: Test3: Projection on w for different categories. The red one is classical music; the black one is country music; the blue one is rock band music.

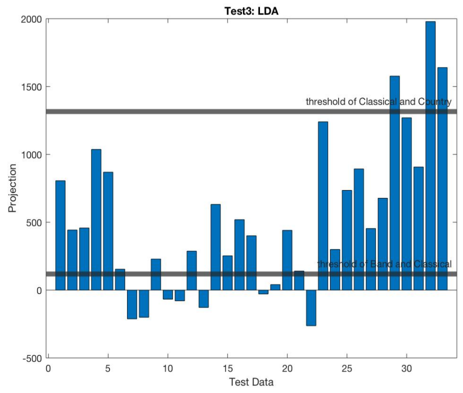

Figure 6: Test3: Bar plot of the statistics associated with the three categorical music data once projected onto the LDA basis.

5 Summary and Conclusions

In this homework, we apply the Linear Discriminant Analysis of three different categorical music to classify the music. Although we don't have fairly high accuracy, from the plots in the computational results section, we could still tell the difference between different categorical music. The first test has the highest accuracy of 63.64%. This could because the training and testing music are from the same artists. The second test has an accuracy of 60.61% and the third test has an accuracy of 42.42%. We also have the plots showing the projection of the training dataset on a projection vector w.The projection plots(Figure(1,3,5)) looks pretty good. However, the projection of the testing dataset on the LDA basis is not that good. In the future, we could future investigate what kind of criterion would affect accuracy.

Appendix A.MATLAB functions used[6]

1. svd(): performs a singular value decomposition of matrix A, such that A = U*S*V'.
2. reshape(): reshape(A,sz) reshapes A using the size vector, sz, to define size(B).
3. sort(): sort the data.
4. spectrogram(): 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.

[6]Other functions applied in this report can be found in the first report. The full description of each function below can be foun on : "MATLAB." MATLAB Documentation, www.mathworks.com/help/matlab/.

Appendix B. MATLAB codes

All cases share the same method, but you need to change the width and range of the filer and load different files. Below is the code for test1.

```
clear all; close all; clc;

%% train dataset
train=[];
for str=["pop","kpop","Beethoven"]
    for i=1:2
        [y, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        y=y'/2;
        y=y(1,:)+y(2,:); % convert to one sound link
        for j=10:5:100
            five=y(Fs*j:Fs*(j+5));
            five = spectrogram(five);
            fivespec=abs(five);
            fivespec=reshape(fivespec, [1,32769*8]);
            train=[train; fivespec];
        end
    end
end
train=train';
sz = size(train,2)/3;

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

mpop = mean(pop,2);
mkpop = mean(kpop,2);
mb = mean(Beethoven,2);
m = (mpop+mkpop+mb)/3;

Sw = 0; % within class variances
for k=1:sz
    Sw = Sw + (pop(:,k)-mpop)*(pop(:,k)-mpop)';
end
for k=1:sz
    Sw = Sw + (kpop(:,k)-mkpop)*(kpop(:,k)-mkpop)';
end
for k=1:sz
    Sw = Sw + (Beethoven(:,k)-mb)*(Beethoven(:,k)-mb)';
end

Sb = ((mpop-m)*(mpop-m)'+(mkpop-m)*(mkpop-m)'+(mb-m)*(mb-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'*pop;
vkpop = w'*kpop;
vBeethoven= w'*Beethoven;

%% find threshold

% vpop < threshold1 < Beethoven < threshold2 < vkpop

sortpop = sort(vpop);
sortkpop = sort(vkpop);
sortBeethoven=sort(vBeethoven);

plot(sortpop,zeros(38),'ob');
hold on
plot(sortkpop,ones(38),'dr');
plot(sortBeethoven,1/2*ones(38),'k*');
% xline(threshold1,'-','threshold of Beethoven and Jackson','Linewidth',5);
% xline(threshold2,'-','threshold of Beethoven and Jackson','Linewidth',5);
title('Test1: Porjection on w','Fontsize',16)
ylabel('Category','Fontsize',16)
xlabel('Porjection on w','Fontsize',16)


t1 = length(sortpop);
t2 = 1;
while sortpop(t1)>sortBeethoven(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold1 = (sortpop(t1)+sortBeethoven(t2))/2;

t3=length(sortBeethoven);
t4=1;
while sortBeethoven(t3)>sortkpop(t4)
    t3=t3-1;
    t4=t4+1;
end
threshold2 = (sortBeethoven(t3)+sortkpop(t4))/2;

%% test data
test=[];
for i=3
    for str=["pop","kpop","Beethoven"]
        [y, Fs]=audioread(strcat(str{1},num2str(i),".mp3"));
        y=y'/2;
        y=y(1,:)+y(2,:);
        for j=50:10:150 % 11 piece from every music
            five=y(Fs*j:Fs*(j+5));
            five = spectrogram(five);
            fivespec=abs(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('Test1: LDA')
hold on;
yline(threshold1,'-','threshold of Beethoven and Jackson','Linewidth',5);
hold on;
yline(threshold2,'-','threshold of EXO and Beethoven','Linewidth',5);
hold off;

p=pval;
for i=1:33
    if p(i)<threshold1
        p(i)=-1;%pop
    elseif  p(i)>threshold2
        p(i)=1;%kpop
    else threshold1 < p(i) < threshold2
        p(i)=0;%Beethoven
    end
end

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

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

correct=c/33
```