Music Classification with LDA

Author: Kuan-Wei Lee

Abstract:

The goal of this project is to classify different genres of music with Linear Discriminant Analysis (LDA). Linear Discriminant Analysis involves techniques like SVD (Singular Value Decomposition) and machine learning. The algorithm I develop on MatLab will take several five seconds music clips as training set and use LDA to differentiate the features of the music I chose.

Sec I.

Introduction

In this project, I selected some music produced by three artists that are distinct in styles. The artists I chose was: Adel, Linkin Park and Bach. I selected 10 songs from each artists as my training set and use LDA to pick out the distinct features of the songs made by these three artists. After the distinct features and the threshold are found, I selected three songs from each artists that was not originally in the training set as the test of my classification algorithm.

Sec II.

Theoretical Background

Linear Discriminant analysis:

The idea of the LDA is simply finding a projection that maximize the inter-class data while minimize the distance between the intra-class data. Finding such a projection involves solving a eigenvalue problem:

$$\widehat{\Sigma_b}\Phi = \lambda\widehat{\Sigma_w}\Phi \qquad (1)$$

$$\Sigma_b = \sum_{i=1}^{n} m_i(\overline{x_i} - \overline{x})(\overline{x_i} - \overline{x})' \qquad (2)$$

$$\Sigma_w = \sum_{i=1}^{n} \sum_{x\in c_i}(\overline{x_i} - \overline{x})(\overline{x_i} - \overline{x})' \qquad (3)$$

Σ_b is the between class variance scatter matrix
Σ_w is the with-in class variance scatter matrix

The projection that gives the biggest separation between inter-class data is the row of Φ that corresponds to the biggest eigenvalue.

Sec III.

Algorithm Implementation and Development

The implementation of my algorithm starts with obtaining music files. I converted the youtube videos of the music I selected to mp3 files and use the program “Audacity” to extract five seconds clips of the music files and save them as .wav files. After I obtained a few five seconds music clips, I loaded the files into MATLAB and use the commend “spectrogram” to get the spectrogram of all the files. I reshape all the spectrograms into a column vector and concatenate the vectors into a single matrix. I perform the SVD on the matrix and perform n-rank approximation since I expect not all the features of the matrix are important when we are classifying the music. After I did the n-rank approximation of the original matrix, I then calculated the with-in and between class variance scatter matrix. The projection that makes the separation between the inter-class data was found by solving the eigenvalue problem and select the vector that correspond to the biggest eigenvalue. After I found the projection, I project each class of my data on that axis and define thresholds. After the thresholds were obtained, I then select some music that were not in my training data to test if the algorithm did a good job on classifying music.

Sec IV.

Computational Results

After performing the LDA on the data and project the data on the axis that makes the separation between inter-class data the greatest, there are clear separation between data (Figure1) . Since the sample sizes of my data is not very big, I set my thresholds by eyeballing it (Threshold 1 is 45 and Threshold 2 is 100). When I project the test data on the axis of greatest separation, any values that is smaller than 45 will be Bach’s music, between 45 and 100 will be Linkin Park’s music and greater than 100 will be Adele’s music. The order of my test data is that the first three songs are Adele’s songs, fourth to the sixth are Bach’s songs, and seventh to the last are Linkin Parks song. The answer given by the classification algorithm is that the first song is Adele’s song, second is Linkin Parks’, third to the sixth is Back’s, and seventh to the last is Adele’s song. As we can see, the algorithm is far from perfect. There are several reasons that caused the algorithm failed to classify the music correctly. The first obvious reason is that the sample size is not big enough. If the training set is not big enough, the algorithm will have a hard time picking out the features that set those music apart. Other reasons may be that a specific artist chosen have a wide range of style that may overlaps with the music of the other artists’. A highly plausible flaws of this algorithm is that it only take five seconds of music clips which are very short. Five seconds of music may not contain enough distinct features to set each classes apart.

Sec V.

Conclusion and Summary

LDA is indeed a powerful tool when it comes to classification. Finding the projection of the data that separates individual classes apart with techniques of linear algebra is truly amazing. However, these techniques will not achieve good results without large sample sizes and carefully chosen thresholds.

Appendix A.

```
function [U,S,V,w,sort1,sort2,sort3] =
music_trainer(mus1,mus2,mus3,feature)
    nd = size(mus1,2);

    [U,S,V] = svd([mus1 mus2 mus3],'econ');

    musics = S*V'; % projection onto principal components
    U = U(:,1:feature);
    class1 = musics(1:feature,1:nd);
    class2 = musics(1:feature,nd+1:nd+nd);
    class3 = musics(1:feature,2*nd+1:end);

    m1 = mean(class1,2);
    m2 = mean(class2,2);
    m3 = mean(class3,2);

    Sw = 0; % within class variances

    for k=1:nd
        Sw = Sw + (class1(:,k)-m1)*(class1(:,k)-m1)';
    end

    for k=1:nd
        Sw = Sw + (class2(:,k)-m2)*(class2(:,k)-m2)';
    end

    for k=1:nd
        Sw = Sw + (class3(:,k)-m3)*(class3(:,k)-m3)';
    end
    x_bar = (nd*m1+nd*m2+nd*m3)/(3*nd);
    Sb = nd*(m1-x_bar)*(m1-x_bar)'; % between class
    Sb = Sb + nd*(m2-x_bar)*(m2-x_bar)';
    Sb = Sb + nd*(m3-x_bar)*(m3-x_bar)';

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

    v1 = w'*class1;
    v2 = w'*class2;
    v3 = w'*class3;


    sort1 = sort(v1);
    sort2 = sort(v2);
    sort3 = sort(v3);

end
```

The above function is the algorithm that performs LDA and find the data projection that separate inter-class data the most

```
[U1,S1,V1] = svd(X1,'econ');
```

The above function is the SVD. It performs singular values decomposition to the matrix X.

Appendix B.

```
clear all
close all
clc

files = dir(fullfile('Adele','*.wav'));
cd Adele
for i = 1:length(files)
    [y,Fs] = audioread(files(i).name);
    y = mean(y,2);
    y = y(1:10:end);
    [s,w,t] = spectrogram(y,500,250);
    Adele(:,i) = s(:);
end
cd ../

files = dir(fullfile('Linkinp','*.wav'));
cd Linkinp
for i = 1:length(files)
    [y,Fs] = audioread(files(i).name);
    y = mean(y,2);
    y = y(1:10:end);
    [s,w,t] = spectrogram(y,500,250);
    Linkp(:,i) = s(:);
end
cd ../
```

```
files = dir(fullfile('Bach','*.wav'));
cd Bach
for i = 1:length(files)
    [y,Fs] = audioread(files(i).name);
    y = mean(y,2);
    y = y(1:10:end);
    [s,w,t] = spectrogram(y,500,250);
    Bach(:,i) = s(:);
end
cd ../

musclass = [Adele Linkp Bach];

%%
clear all
close all
clc

files = dir(fullfile('test1','*.wav'));
cd test1
for i = 1:length(files)
    [y,Fs] = audioread(files(i).name);
    y = mean(y,2);
    y = y(1:10:end);
    [s,w,t] = spectrogram(y,500,250);
    test1(:,i) = s(:);
end
cd ../
save('test1.mat','test1')


%%
clear all
close all
clc

load('Adele.mat')
load('Linkp.mat')
load('Bach.mat')

feature = 10;
[U,S,V,w,sort1,sort2,sort3] =
music_trainer(abs(Adele),abs(Linkp),abs(Bach),feature);

plot(sort1,zeros(1,length(sort1)),'bo','LineWidth',2)
hold on
% histogram(sort2,20)
plot(sort2,zeros(1,length(sort1)),'ro','LineWidth',2)
% histogram(sort3,20)
plot(sort3,zeros(1,length(sort1)),'ko','LineWidth',2)
legend('Adele','Linkin Park','Bach')
```

```
title('Projection of data')
threshold1 = 45;
threshold2 = 100 ;
y = -5:0.01:5;
plot(threshold1*ones(1,length(y)),y,'r-')
plot(threshold2*ones(1,length(y)),y,'m-')

%%
load('test1.mat')

testmat = U'*abs(test1);
pval = w'*testmat;

answer = [1 1 1 3 3 3 2 2 2];

classify = zeros(1,length(pval));

classify(pval<=threshold1) = 3;

classify((threshold1<=pval)&(pval<=threshold2)) = 2;
classify(pval>=threshold2) = 1;
```