Music Classification

Khoa Vo

March 7, 2020

Abstract

This project will discuss and perform one of the techniques to do classification problem, Linear Discrimination Analysis. We will try to classify the music of three different artists/genres in 3 cases after projecting onto LDA basis. The accuracy rate will be reported to see how well this technique does.

1 Introduction and Overview

Music genre is effortlessly recognized by human's brain. We humans are astoundingly good at making sense of what we are hearing, but nearly all that work is done unconsciously, therefore, we usually don't appreciate how our brain works. Music Classification is a complex problem to be programmed.There are a few techniques to approach this problem, one of those are Linear Discrimination Analysis.
This project will go through the process of building a program that classify music in three different cases: 3 different bands/singers, 3 different bands/singers having the same genre, and 3 different genres. The program will be tested with a number of 5-second clips of music with the labeled singers and genres. The accuracy rates of the program will be shown.

2 Theoretical Background

In order to classify music, we need to find the principal components by using SVD (Section 2.1) associated with each given singer/band/genre data. Then, designing a statistical decision threshold for discrimination among those singers/bands/genres (Section 2.2).

2.1 Singular Value Decomposition (SVD)

Singular Value Decomposition is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in application. Specifically, a $m \times n$ matrix $\mathbf{A}$ has Singular Value Decomposition in form of $\mathbf{U\Sigma V}^*$, where $\mathbf{U}$ is a $m \times m$ unitary, $\mathbf{\Sigma}$ is $m \times n$ diagonal matrix with non-negative real numbers σ_i , and $\mathbf{V}$ is a $n \times n$ unitary matrix, i.e., SVD decomposes matrix $\mathbf{A}$ into three transformation: an initial rotation $\mathbf{V}^*$, a scaling $\mathbf{\Sigma}$, and a final rotation $\mathbf{U}$. The singular values σ_i are uniquely determined, and, if matrix $\mathbf{A}$ is square and the σ_i distinct, the singular vectors u_j and v_j are uniquely determined up to complex signs. The SVD can be computed as the following:

$$\begin{aligned}\mathbf{A}^T\mathbf{A} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*) \\ &= \mathbf{V}\mathbf{\Sigma}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \\ &= \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^* \end{aligned} \tag{1}$$

and

$$\begin{aligned}\mathbf{A}\mathbf{A}^T &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^*)^T \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*\mathbf{V}\mathbf{\Sigma}\mathbf{U}^* \\ &= \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^* \end{aligned} \tag{2}$$

Multiplying (1) and (2) by $\mathbf{V}$ and $\mathbf{U}$ respectively gives two self-consistent eigenvalue problems.

$$\mathbf{A}^T\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{\Sigma}^2 \tag{3}$$

$$\mathbf{A}\mathbf{A}^T\mathbf{U} = \mathbf{U}\mathbf{\Sigma}^2 \tag{4}$$

If eigenvectors are found, then the orthonormal basis vectors are produced for $\mathbf{U}$ and $\mathbf{V}$. Also, SVD diagonalizes and each singular direction captures highest energy possible. The percentage of energy captured by σ_i can be calculated by:

$$E_i = \frac{{\sigma_i}^2}{{\sigma_1}^2 + {\sigma_2}^2 + ... + {\sigma_r}^2}, \text{ with } r \text{ is the rank of matrix } \mathbf{A} \tag{5}$$

2.2 Linear Discrimination Analysis

In order to make the final decision, the trained data sets are projected onto a new basis. To achieve high accuracy rate, the data should be well separated, not mixed, when projected onto new basis. The "best" subspace is a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. A projection $\mathbf{w}$ can be constructed as

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \tag{6}$$

where between-class $\mathbf{S}_B$ and within-class $\mathbf{S}_W$ can be defined as

$$\mathbf{S}_B = \sum_{j=1}^{n} m_j(\vec{u}_j - \vec{u})(\vec{u}_j - \vec{u})^T \tag{7}$$

$$\mathbf{S}_W = \sum_{j=1}^{n} \sum_{\vec{x}} (\vec{x} - \vec{u}_j)(\vec{x} - \vec{u}_j)^T \tag{8}$$

3 Algorithm Implementation and Development

3.1 Data processing

For the purpose of this project, we need two different data sets, one for training and one for testing. We will perform the music classification on 3 different cases, however, the process of obtaining data for training and testing is similar. For training data, I will choose 4 music pieces from each artist/genre and extract 6 5-second clips each from those pieces. For testing data, I will do the same, but with different time in those music pieces. Thus, I will have 72 5-second clips for training and 72 5-second clips for testing.

The data music for each clip loaded into Matlab will be stereo, which has two columns, I will take the first column to make it mono to analyze. We want the spectrum of frequencies of the clip as it varies with time, so we use the built-in function "spectrogram" in Matlab to capture the spectrogram, flatten it into one column, and take the real part of it. Then, we center the data by subtracting the mean of each column to perform SVD later.

3.2 Training Function - Linear Discrimination Analysis

First, we want to perform SVD on the data we processed and find the projection of the data onto the principal components by calculating $S * V^T$. We take a number of features of the principal component projection of the data of each artist/genre, we need to figure out which value of feature parameter works the best. Then, we calculate between-class variances and within-class variances by using equation (7) and equation (8). Then, we perform Linear Discriminant Analysis by perform equation (9), in which we find the eigenvalues and eigenvectors of $\mathbf{S}_B$, $\mathbf{S}_W$, determine the maximal eigenvalue and corresponding eigenvector, and finally we find new basis, $\mathbf{w}$, by normalize the largest eigenvector. Finally, we will want to return new basis $\mathbf{w}$, a list of values that each artist/genre is projected onto this new basis in ascending order, and $\mathbf{U}$ with first 30 features.

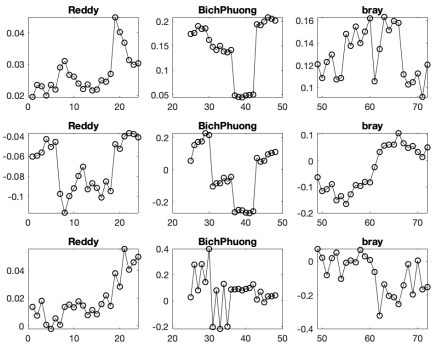
3.3 Define The Threshold to Classify

The decision threshold is computed based on how the training data is projected onto $\mathbf{w}$. Since we have 3 different artist/genre to classify, therefore, we will need 2 decision thresholds, one for artist/genre has the least mean artist/genre has the second largest mean and the other one for artist/genre has the second largest mean artist/genre has the largest mean.

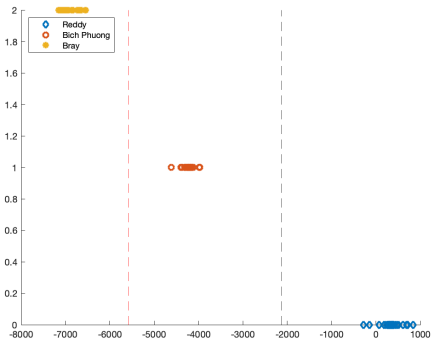
To compute the threshold, we will start with the largest value of artist/genre has smaller mean, a_n, and the smallest value of artist/genre has larger mean, b_1. While $a_n > b_1$, we will move another two points, a_{n-1} and b_2, until $a < b$. The threshold will be the average of a and b. This algorithm will ensure that the decision thresholds are unbiased when we perform the classification.

3.4 Classification

We first perform PCA projection will the testing data and then LDA projection. Then compare the value we got with the thresholds to classify. The hidden label of testing data is provided to compute the accuracy rate of the program.

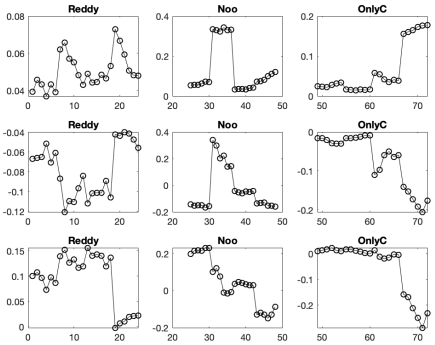


(a) Projection of the first 24 5-second clips of each artist onto the first three POD modes

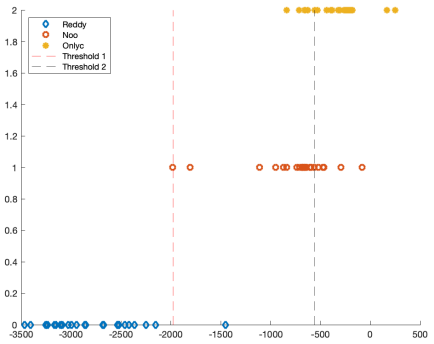


(b) Trained data projected on the LDA basis

Figure 1: Trained Data Projection onto first 3 POD modes and LDA basis



(a) Projection of the first 24 5-second clips of each artist onto the first three POD modes



(b) Trained data projected on the LDA basis

Figure 2: Trained Data Projection onto first 3 POD modes and LDA basis

4 Computational Results

4.1 Case 1: Band Classification

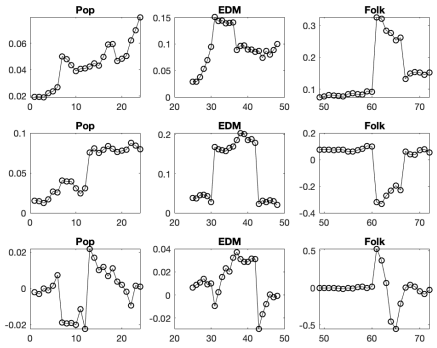
In this case, I use music of Reddy (Pop/Ballad), Bich Phuong (Dance), and B-ray(Hiphop/Rap). These 3 artists play 3 different genres.

The accuracy rate for this case is 97.22%, the feature parameter is set to be 42. The thresholds are -5576 and -2124. From Figure 1.b, we can see that there are no overlap among these artists, this is why we can get a high accuracy rate.

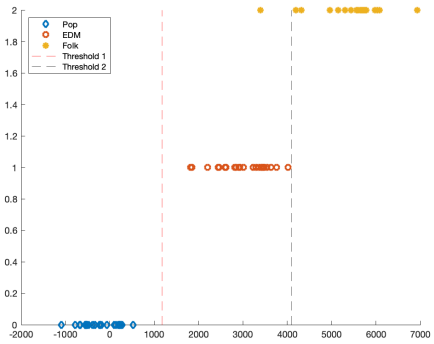
4.2 Case 2: The Case for Seattle

In this case, I use 3 different artists in Pop/Ballad: Reddy, Noo, and OnlyC.

The accuracy rate for this case is 69.44% with the feature parameter is set to be 30. The thresholds are -1978 and -559. From Figure 2.b, we can see that there is not much overlap between Reddy and Noo, however, there is a large overlap between Noo and Only C. Since these three artist have the same genre of music, hence data projected onto LDA basis is expected to have some overlaps, this is why performing classification does not have a high accuracy rate as in case 1.



(a) Projection of the first 24 5-second clips of each artist onto the first three POD modes



(b) Trained data projected on the LDA basis

Figure 3: Trained Data Projection onto first 3 POD modes and LDA basis

4.3 Case 3: Genre Classification

In this case, I use music of 3 genres: Pop, Electronic dance music (EDM), and Folk.

The accuracy is 63.89% with the feature parameter is set to be 30. The thresholds are 1173 and 4103. Figure 3.b indicates that there is not an large overlap among these genres, but this case has the lowest accuracy rate. It might be because the high variability within each class of the testing data.

5 Summary and Conclusions

Linear Discrimination Analysis is one of techniques that can be used to do classification problems. In this project, the accuracy rates for music classification were decreased as the cases getting more complicated. One factor that could affect the result is that the way we choose and process our data. If we can collect more data for training, it will potentially increase the accuracy rate for classification.

Appendix A

svd(): performs a singular value decomposition of matrix.
spectrogram(): returns the short-time Fourier transform of the input signal.

Appendix B

Code to generate training testing data

```
clear all; close all; clc
reddy = ["VaiGiayNuaThoi.mp3" "Neu.mp3" "Gui.mp3" "ThiThoi.mp3"];
noo = ["TELDAKTN.mp3" "NhuPhutBanDau.mp3" "NhungKeMongMo.mp3"
    "ThuongMayCungLaNguoiDung.mp3"];
onlyc = ["YeuLaThaThu.mp3" "YeuDiDungSo.mp3" "GoiTenCoDon.mp3" "AnhDaSai.mp3"];
bray = ["CaoOc20.mp3" "PhepMau.mp3" "ConTraiCung.mp3" "BSNL.mp3"];
bichphuong = ["BuiYeu.mp3" "BaoGioLayChong.mp3" "GuiAnhXaNho.mp3"
    "DiDuDuaDi.mp3"];
pop = ["YeuLaThaThu.mp3" "ThiThoi.mp3" "TELDAKTN.mp3"
    "ThuongMayCungLaNguoiDung.mp3"];
edm = ["ChungTaKhongThuocNhau.mp3" "ThuCuoi.mp3" "TuyAm.mp3" "SongGio.mp3"];
folk = ["folk1.mp3" "folk2.mp3" "folk3.mp3" "fold4.mp3" ];
TestSet = [];
label = [];
for i = 1:4
    [y, Fs] = audioread(pop(i));
    y = y(:,1); % stereo to mono
    [m,n]=size(y);
    dt=1/Fs;
    t=dt*(0:m-1);
    idx1 = (t>36) & (t<41);
    idx2 = (t>37) & (t<42);
    idx3 = (t>38) & (t<43);
    idx4 = (t>39) & (t<44);
    idx5 = (t>40) & (t<45);
    idx6 = (t>41) & (t<46);
    y1 = y(idx1);
    y1 = y1(1:220499, 1);
    y1 = abs(spectrogram(y1));
    y1 = y1(:);
    y1 = y1 - mean(y1);
    y2 = y(idx2);
    y2 = y2(1:220499, 1);
    y2 = abs(spectrogram(y2));
    y2 = y2(:);
    y2 = y2 - mean(y2);
    y3 = y(idx3);
    y3 = y3(1:220499, 1);
    y3 = abs(spectrogram(y3));
    y3 = y3(:);
    y3 = y3 - mean(y3);
    y4 = y(idx4);
    y4 = y4(1:220499, 1);
    y4 = abs(spectrogram(y4));
    y4 = y4(:);
    y4 = y4 - mean(y4);
    y5 = y(idx5);
    y5 = y5(1:220499, 1);
    y5 = abs(spectrogram(y5));
```

```
    y5 = y5(:);
    y5 = y5 - mean(y5);
    y6 = y(idx6);
    y6 = y6(1:220499, 1);
    y6 = abs(spectrogram(y6));
    y6 = y6(:);
    y6 = y6 - mean(y6);
    label = [label 0 0 0 0 0 0];
    TestSet = [TestSet y1 y2 y3 y4 y5 y6];
end
for i = 1:4
    [y, Fs] = audioread(edm(i));
    y = y(:,1); % stereo to mono
    [m,n]=size(y);
    dt=1/Fs;
    t=dt*(0:m-1);
    idx1 = (t>36) & (t<41);
    idx2 = (t>37) & (t<42);
    idx3 = (t>38) & (t<43);
    idx4 = (t>39) & (t<44);
    idx5 = (t>40) & (t<45);
    idx6 = (t>41) & (t<46);
    y1 = y(idx1);
    y1 = y1(1:220499, 1);
    y1 = abs(spectrogram(y1));
    y1 = y1(:);
    y1 = y1 - mean(y1);
    y2 = y(idx2);
    y2 = y2(1:220499, 1);
    y2 = abs(spectrogram(y2));
    y2 = y2(:);
    y2 = y2 - mean(y2);
    y3 = y(idx3);
    y3 = y3(1:220499, 1);
    y3 = abs(spectrogram(y3));
    y3 = y3(:);
    y3 = y3 - mean(y3);
    y4 = y(idx4);
    y4 = y4(1:220499, 1);
    y4 = abs(spectrogram(y4));
    y4 = y4(:);
    y4 = y4 - mean(y4);
    y5 = y(idx5);
    y5 = y5(1:220499, 1);
    y5 = abs(spectrogram(y5));
    y5 = y5(:);
    y5 = y5 - mean(y5);
    y6 = y(idx6);
    y6 = y6(1:220499, 1);
    y6 = abs(spectrogram(y6));
    y6 = y6(:);
    y6 = y6 - mean(y6);
    label = [label 1 1 1 1 1 1];
```

```
    TestSet = [TestSet y1 y2 y3 y4 y5 y6];
end

for i = 1:4
    [y, Fs] = audioread(folk(i));
    y = y(:,1); % stereo to mono
    [m,n]=size(y);
    dt=1/Fs;
    t=dt*(0:m-1);
    idx1 = (t>36) & (t<41);
    idx2 = (t>37) & (t<42);
    idx3 = (t>38) & (t<43);
    idx4 = (t>39) & (t<44);
    idx5 = (t>40) & (t<45);
    idx6 = (t>41) & (t<46);
    y1 = y(idx1);
    y1 = y1(1:220499, 1);
    y1 = abs(spectrogram(y1));
    y1 = y1(:);
    y1 = y1 - mean(y1);
    y2 = y(idx2);
    y2 = y2(1:220499, 1);
    y2 = abs(spectrogram(y2));
    y2 = y2(:);
    y2 = y2 - mean(y2);
    y3 = y(idx3);
    y3 = y3(1:220499, 1);
    y3 = abs(spectrogram(y3));
    y3 = y3(:);
    y3 = y3 - mean(y3);
    y4 = y(idx4);
    y4 = y4(1:220499, 1);
    y4 = abs(spectrogram(y4));
    y4 = y4(:);
    y4 = y4 - mean(y4);
    y5 = y(idx5);
    y5 = y5(1:220499, 1);
    y5 = abs(spectrogram(y5));
    y5 = y5(:);
    y5 = y5 - mean(y5);
    y6 = y(idx6);
    y6 = y6(1:220499, 1);
    y6 = abs(spectrogram(y6));
    y6 = y6(:);
    y6 = y6 - mean(y6);
    label = [label 2 2 2 2 2 2];
    TestSet = [TestSet y1 y2 y3 y4 y5 y6];
end
save('test_data', 'TestSet', 'label')
```

Code to perform classification

```
clear all; close all; clc

load('test_data.mat');
load('train_data.mat');


%% prepare testset

feature = 30;
[U,S,V,w,pop,EDM,folk] = dc_trainer(data1,data2,data3, feature);



mean(pop);
mean(EDM);
mean(folk);


TestMat = U'*TestSet; % PCA projection
pval = w'*TestMat; % LDA projection

figure(2)
for k=1:3
    subplot(3,3,3*k-2)
    plot(1:24,V(1:24,k),'ko-'), title('Pop')
    subplot(3,3,3*k-1)
    plot(25:48,V(25:48,k),'ko-'), title('EDM')
    subplot(3,3,3*k)
    plot(49:72,V(49:72,k),'ko-'), title('Folk')
end

t1 = length(pop);
t2 = 1;
while pop(t1)>EDM(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold1 = (pop(t1)+EDM(t2))/2
t1 = length(EDM);
t2 = 1;
while EDM(t1)>folk(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold2 = (EDM(t1)+folk(t2))/2
count = 0;
for i = 1:size(pval, 2)
    if pval(i) < threshold1
        if label(i) == 0
            count = count + 1;
        end
```

```
    elseif pval(i) > threshold2
        if label(i) == 2
            count = count + 1;
        end
    else
        if label(i) == 1
            count = count + 1;
        end
    end
end
accuracy_rate = count/size(label,2)

figure(1)
twos = zeros(1, size(folk, 2)) + 2;
%h = plot(reddy, zeros(size(reddy, 2)),'dr',noo, ones(size(noo, 2)),'ob',onlyc,
    twos, '*g','Linewidth',2)

scatter(pop, zeros(1, size(pop, 2)),'d', 'Linewidth', 2)
hold on
scatter(EDM, ones(1, size(EDM,2)),'o', 'Linewidth', 2)
scatter(folk, twos, '*', 'Linewidth', 2)
xline(threshold1,'--r');
xline(threshold2,'--k');
legend('Pop', 'EDM', 'Folk','Threshold 1', 'Threshold 2', 'Location',
    'northwest')
```

```
%%MUSIC_TRAINER
function [U,S,V,w,sortelec,sortinst, sorthhop] =
    dc_trainer(elec,inst,hhop,feature)
    [m1, n1] = size(elec);
    [m3, n2] = size(inst);
    [m3, n3] = size(hhop);
    [U,S,V] = svd([elec inst hhop],'econ');

    music = S*V'; % projection onto principal components
    U = U(:,1:feature);
    elecs = music(1:feature,1:n1);
    insts = music(1:feature,n1+1:n1+n2);
    hhops = music(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(elecs,2);
    m2 = mean(insts,2);
    m3 = mean(hhops,2);
    m = (m1 + m2 + m3)/3;

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (elecs(:,k)-m1)*(elecs(:,k)-m1)';
    end
```

```
    for k=1:n2
        Sw = Sw + (insts(:,k)-m2)*(insts(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (hhops(:,k)-m3)*(hhops(:,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);

    velec = w'*elecs;
    vinst = w'*insts;
    vhhop = w'*hhops;

    sortelec = sort(velec);
    sortinst = sort(vinst);
    sorthhop = sort(vhhop);

end
```