Linear Discriminant Analysis on Music Classification

Htet Aung Myin

March 6, 2020

Abstract

The application of the Linear Discriminant Analysis on different audios of different artists and genre. Three tests are performed. The first test involves classification between three different bands and genre. The second test repeats the first test with the exception of three bands from the same genre. The third test is where the classification is performed on three songs from different genres.

1 Introduction and Overview

The problem given is regarding three tests situations. In the first test, three different bands and genres were identified based on their unique aspects. The second test repeats the first test except the classification was performed on three different bands of the same genre. In the third test, the classification is performed on the three differnet genres.

2 Theoretical Background

2.1 Singular Value Decomposition

The Singular Value Decomposition transforms a matrix A into:

$$\begin{aligned} \mathbf{A} &= \mathbf{U\Sigma V}^* \text{ where} \\ \mathbf{U} &\in \mathbb{C}^{m\times m} \text{ is unitary} \\ \mathbf{V} &\in \mathbb{C}^{n\times n} \text{ is unitary} \\ \mathbf{\Sigma} &\in \mathbb{R}^{m\times n} \text{ is diagonal} \end{aligned}$$

The SVD can be computed by the following

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

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

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

2.2 Principal Component Analysis

One of the primary applications of the SVD is for Principal Component Analysis (PCA)where it reduces the dimensions of the data set. The data can be placed a matrix with $X \in \mathbb{R}^{m\times n}$, where m is number of measurements and n is number of data points taken over time. In this matrix, redundancy can be identified and reduced with the covariance matrix and is defined by:

2.3 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) is used to find a linear combination of features that characterizes of separates two or more classes of objects in the data. For a two class LDA, the projection w is:

$$w = argmax \frac{w^T S_B w}{w^T S_W w}$$

and S_B and S_W are given by:

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

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

3 Algorithm Implementation and Development

Audio data is first clipped into 5 second clips using an external tool. The clipped samples are then converted from .mp3 to .mat.

1. The audios are converted to mono by averaging the two vectors together. Each .mat file are clipped to the length of the smallest vector to ensure they are all the same size.

2. Training data and testing data is split with a ratio of 70

3. Spectrograms are made for both training and test data. Each matrix formed is then reshaped into a singular vector for each test case.

4. A two class LDA is attempted to compare between each audio samples for both the training and test data.

5. This process is the repeated over all three tests with comparison between their specific features.

6. Note: As a 3 class LDA algorithm was not successfully implemented, a 2 class LDA algorithm is used to compare two songs instead. Each threshold from the LDA is set manually as threshold calculated from the mean made the test data statically insignificant. A modification is done by only comparing two variables per test in order to complete this report.

4 Computational Results

There are none available as the LDA algorithm is not working properly. A statistically relevant threshold cannot be found. There may be issues with transforming data for proper analysis.

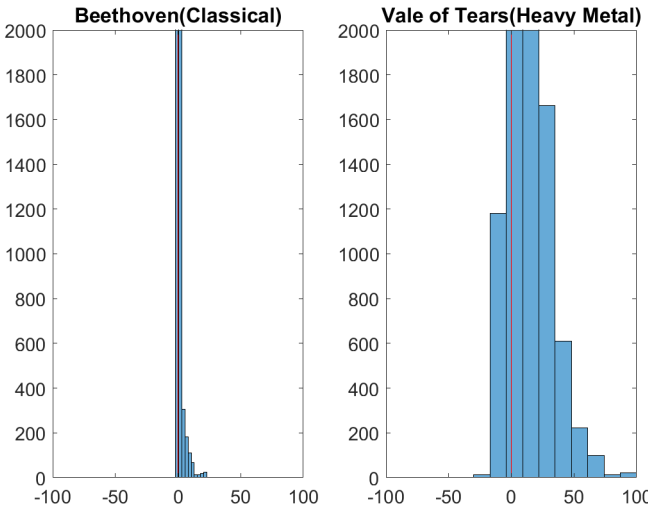

Figure 1: Test 1/3: Comparing Two Different Genres and Music

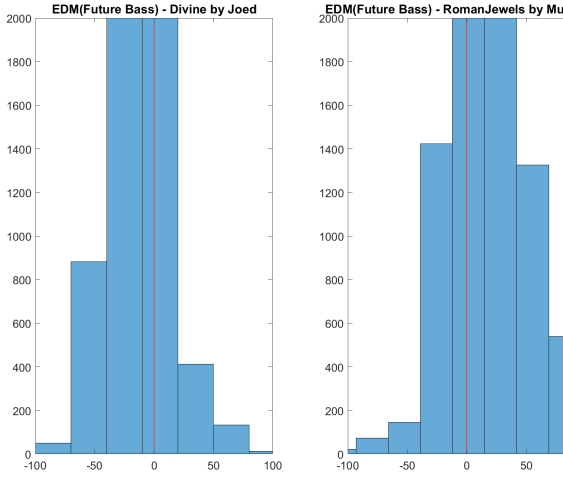

Figure 2: Test 2: Comparing Music From Same Genre

5 Summary & Conclusion

As the LDA algorithm was not working properly, a statistical threshold could not be found. Possible errors could be from converting audio to wavelets and incorrect implementation of the LDA.

6 Appendix A.

- diag(X) = returns diagonal value of matrix X
- find(X) = returns a vector of index in a matrix
- zeros() = fills a matrix with zeros
- svd() = returns a diagonal matrix S along with unitary matrix U and V
- size(X) : returns dimension of matrix X
- length(X) : returns length of matrix X

7 Appendix B.

```
%% convert files to .mat format
clear; close all; clc

filenames = {'beethoven-moonlightsonata.mp3', 'Deflo &
    Lliam Taylor - Spotlight (feat. AWA).mp3', 'divine-
    joed.mp3','peculate-valeoftears.mp3','romanJewels.
    mp3','sadpuppy-youhavetheblame.mp3'};


filename = filenames{1};
[y1,Fs1] = audioread(filename);
y1 = mean(y1,2);
save m1 y1 Fs1
clear filename

filename = filenames{2};
[y2,Fs2] = audioread(filename);
y2 = mean(y2,2);
save m2 y2 Fs2
clear filename
filename = filenames{3};
[y3,Fs3] = audioread(filename);
y3 = mean(y3,2);
save m3 y3 Fs3
clear filename
filename = filenames{4};
[y4,Fs4] = audioread(filename);
y4 = mean(y4,2);
save m4 y4 Fs4
clear filename
filename = filenames{5};
[y5,Fs5] = audioread(filename);
y5 = mean(y5,2);
save m5 y5 Fs5
clear filename
filename = filenames{6};
[y6,Fs6] = audioread(filename);
y6 = mean(y6,2);
```

```
36 save m6 y6 Fs6
37 clear filename
38
39 [trainy1, trainFs1] = audioread('TrainMoonlight.mp3');
40 trainy1 = mean(trainy1,2);
41 save trainm1 trainy1 trainFs1
42
43 [trainy4, trainFs4] = audioread('TrainVale.mp3');
44 trainy4 = mean(trainy4,2);
45 save trainm4 trainy4 trainFs4
46
47 [trainy3, trainFs3] = audioread('TrainDivine.mp3');
48 trainy3 = mean(trainy3,2);
49 save trainm3 trainy3 trainFs3
50
51 [trainy5, trainFs5] = audioread('TrainRomenJewel.mp3');
52 trainy5 = mean(trainy5,2);
53 save trainm5 trainy5 trainFs5
54
55 %% loading mat files
56 clear; close all; clc
57 load m1.mat %beethoven-moonlightsonata.mp3
58 load m2.mat %Deflo & Lliam Taylor - Spotlight (feat.
     AWA).mp3
59 load m3.mat %divine - joed
60 load m4.mat %peculate-valeoftears.mp3
61 load m5.mat %romanJewels.mp3
62 load m6.mat %sadpuppy-youhavetheblame.mp3
63 %p8 = audioplayer(v3,Fs3);
64 %playblocking(p8);
65
66 minLen = length(y1);
67 y2 = y2(1:minLen);
68 y3 = y3(1:minLen);
69 y4 = y4(1:minLen);
70 y5 = y5(1:minLen);
71 y6 = y6(1:minLen);
72
73 %% spectrograms
74 %v1 = train1'; v2 = train2'; v3 = train3'; v4 = train4
       '; v5 = train5'; v6 = train6';
```

```
75 v1 = y1'; v2 = y2'; v3 = y3'; v4 = y4'; v5 = y5'; v6 =
       y6';
76 n = length(v1);
77 L = length(v1)/Fs1;
78 k=(2*pi/(L))*[0:n/2-1 -n/2:-1];
79 ks=fftshift(k);
80 t = (1:length(v1))/Fs1;
81 a = 125;
82 tslide = 0:0.1:length(v1)/Fs1;
83
84
85
86 for j = 1:length(tslide)
87     g = exp(-a*(t-tslide(j)).^2);
88     Sg1 = g.*v1; Sg2 = g.*v2; Sg3 = g.*v3; Sg4 = g.*v4;
           Sg5 = g.*v5; Sg6 = g.*v6;
89     Sgt1 = fft(Sg1); Sgt2 = fft(Sg2); Sgt3 = fft(Sg3);
          Sgt4 = fft(Sg4); Sgt5 = fft(Sg5);Sgt6 = fft(Sg6)
          ;
90
91     Sgt_spec1(j,:) = fftshift(abs(Sgt1));Sgt_spec2(j,:)
          = fftshift(abs(Sgt2));
92     Sgt_spec3(j,:) = fftshift(abs(Sgt3));Sgt_spec4(j,:)
          = fftshift(abs(Sgt4));
93     Sgt_spec5(j,:) = fftshift(abs(Sgt5));Sgt_spec6(j,:)
          = fftshift(abs(Sgt6));
94
95 end
96 %%
97 figure(6)
98 pcolor(tslide,ks,Sgt_spec1.'),
99 shading interp
100 set(gca,'Ylim',[0 2000],'Fontsize',16)
101 colormap(hot)
102 hold on
103 figure(7)
104 pcolor(tslide,ks,Sgt_spec4.'),
105 shading interp
106 set(gca,'Ylim',[0 2000],'Fontsize',16)
107 colormap(hot)
108
```

```
109 %%
110 feature =50;
111 [U,S,V,threshold,w,sortm1,sortm2] = dc_trainer(
      Sgt_spec1,Sgt_spec4,feature);
112 figure(5)
113 subplot(1,2,1)
114 histogram(sortm1,10); hold on, plot([threshold
      threshold],[0 2000],'r')
115 set(gca,'Xlim',[-100 100],'Ylim',[0 2000],'Fontsize'
      ,14)
116 title('Beethoven(Classical)')
117 subplot(1,2,2)
118 histogram(sortm2,10); hold on, plot([threshold
      threshold],[0 2000],'r')
119 set(gca,'Xlim',[-100 100],'Ylim',[0 2000],'Fontsize'
      ,14)
120 title('Vale of Tears(Heavy Metal)')
121 %%
122 load trainm1.mat %beethoven-moonlightsonata.mp3
123 load trainm3.mat %divine - joed
124 load trainm4.mat %peculate-valeoftears.mp3
125 load trainm5.mat %romanJewels.mp3
126
127 %p8 = audioplayer(v3,Fs3);
128 %playblocking(p8);
129 minLen = length(trainy1);
130 trainy3 = trainy3(1:minLen);
131 trainy4 = trainy4(1:minLen);
132 trainy5 = trainy5(1:minLen);
133 v1 = trainy1'; v3 = trainy3';v4 = trainy4';v5 = trainy5
      ';
134 n = length(v1);
135 L = length(v1)/trainFs1;
136 k=(2*pi/(L))*[0:n/2-1 -n/2:-1];
137 ks=fftshift(k);
138 t = (1:length(v1))/Fs1;
139 a = 125;
140 tslide = 0:0.1:length(v1)/Fs1;
141
142
143
```

```
144 for j = 1:length(tslide)
145     g = exp(-a*(t-tslide(j)).^2);
146     trainSg1 = g.*v1;  trainSg3 = g.*v3; trainSg4 = g.*
          v4; trainSg5 = g.*v5;
147     trainSgt1 = fft(Sg1);  trainSgt3 = fft(Sg3);
          trainSgt4 = fft(Sg4); trainSgt5 = fft(Sg5);
148     trainSgt_spec1(j,:) = fftshift(abs(trainSgt1));
149     trainSgt_spec3(j,:) = fftshift(abs(trainSgt3));
150     trainSgt_spec4(j,:) = fftshift(abs(trainSgt4));
151     trainSgt_spec5(j,:) = fftshift(abs(trainSgt5));
152
153 end
154 pval = w'*trainSgt1'
155 %%
156 function [U,S,V,threshold,w,sortm1,sortm2] = dc_trainer
      (m1,m2,feature)
157     nd = size(m1,2); nc = size(m2,2);
158
159     [U,S,V] = svd([m1 m2],'econ');
160
161     music = S*V'; % projection onto principal
          components
162     U = U(:,1:feature);
163     m1 = music(1:feature,1:nd);
164     m2 = music(1:feature,nd+1:nd+nc);
165
166     md = mean(m1,2);
167     mc = mean(m2,2);
168
169     Sw = 0; % within class variances
170     for k=1:nd
171         Sw = Sw + (m1(:,k)-md)*(m1(:,k)-md)';
172     end
173     for k=1:nc
174         Sw = Sw + (m2(:,k)-mc)*(m2(:,k)-mc)';
175     end
176
177     Sb = (md-mc)*(md-mc)'; % between class
178
179     [V2,D] = eig(Sb,Sw); % linear discriminant analysis
180     [~,ind] = max(abs(diag(D)));
```

```
181     w = V2(:,ind); w = w/norm(w,2);
182
183     vm1 = w'*m1;
184     vm2 = w'*m2;
185
186     if mean(vm1)>mean(vm2)
187         w = -w;
188         vm1 = -vm1;
189         vm2 = -vm2;
190     end
191
192
193     sortm1 = sort(vm1);
194     sortm2 = sort(vm2);
195
196     t1 = length(sortm1);
197     t2 = 1;
198     while sortm1(t1)>sortm2(t2)
199         t1 = t1-1;
200         t2 = t2+1;
201     end
202     threshold = (m1(t1)+m2(t2))/2;
203 end
```