

# Music Genre Classification

Jonathan Zhao

February 27th, 2020

## Abstract

This report discusses an algorithm of machine learning to classify different music genres. We will explore the foundations of supervised machine learning and implement a classifier, conducting three tests, and examine the accuracy of our model. To obtain our goal, we will apply the fast Fourier transform, singular value decomposition (SVD), and Linear discrimination analysis (LDA) on a sets of music collected.

## Sec I: Introduction and Overview

Nowadays, machine learning has become widely applied in conducting a variety of jobs of predictions, classifications, recognitions, detections, and recommendations. Some familiar applications may be face recognition technology, traffic predictions, and stock market prediction.

Our goal is to design a code that can classify a given piece of music by sampling a 5 second clip. We will test the functionalities of our classifier, and explore the performance under different training sets and testing sets.

## Sec II: Theoretical Background

### 2.1 Linear Discriminant Analysis

Linear discriminant analysis, or linear discrimination analysis (LDA), is a method used in statistics to find a linear combination of features that characterizes, or intentionally separates, two or more classes of objects. It focuses on maximizing the separability among known classes. LDA can be considered as a dimensionality reduction technique and the results of it can be used for later classification.

The goal of LDA is two-fold: find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. For a two-class LDA, the above idea

results in consideration of the following mathematical formulation. Construct a projection  $\mathbf{w}$  such that

$$\mathbf{w} = \arg \max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} \quad (2.1.1)$$

where

$$\mathbf{S}_B = (\mu_2 - \mu_1) (\mu_2 - \mu_1)^T \quad (2.1.2)$$

$$\mathbf{S}_W = \sum_{j=1}^2 \sum_{\mathbf{x}} (\mathbf{x} - \mu_j) (\mathbf{x} - \mu_j)^T \quad (2.1.3)$$

These quantities essentially measure the variance of the data sets as well as the variance of the difference in the meanings. The solution of the criterion given by 2.1.1 can be found via the generalized eigenvalue problem

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \quad (2.1.4)$$

## 2.2 Multi-class LDA

Multi-class LDA is based on the analysis of two scatter matrices: within-class scatter matrix and between-class scatter matrix. The between-class scatter matrix  $\mathbf{S}_b$  becomes the summation of distances between the center of the total scatters and the center of each category.

$$\mathbf{S}_b = \sum_{k=1}^m n_k (\boldsymbol{\mu}_k - \boldsymbol{\mu}) (\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \quad (2.2.1)$$

where  $m$  is the number of classes,  $\boldsymbol{\mu}$  is the overall sample mean, and  $n_k$  is the number of samples in the  $k$ -th class.

## 2.3 Machine Learning

In general, machine learning is the study of statistical models and algorithms so that computer systems can automatically perform different sorts of tasks without specific instructions or operations on them, but rely on patterns, features, and inference. As a subset of artificial intelligence, it is not only popular in computer science, but also plays a significant role in many other fields, such as bioinformatics and financial industry. The two categories of machine learning are supervised and unsupervised learning. Supervised learning, as the name tells us, needs feeding a labeled datasets and the answers for it to evaluate and to learn on the training data, while unsupervised learning, in contrast, learns on unlabeled datasets and tries to explain the patterns on its own.

In the following, we will implement a music genre classifier, an application of unsupervised learning models to perform item classification based on training pairs.

The major four steps to conduct an image classifier, as present in our course, are:

- 1) Decompose images into wavelet basis functions. - Provide an effective way for doing edge detection.
- 2) Find the principal components associated with the categories respectively from the wavelet expanded images.
- 3) Design a decision threshold for discrimination between categories. The method to be used here will be a linear discrimination analysis we just introduce.
- 4) Test the algorithm efficiency and accuracy by running through testing data.

In the following, we will follow the four steps and make several adjustments, since our objects will be audios rather than images, and the number of categories will be three rather than two. In the following, we will transform the audio file into spectrograms in the first step of wavelet decomposition, and then we will apply SVD to find their principal components just as usual. In the third step, we will search for two thresholds for three categories instead of only one.

## Sec III: Algorithm Implementation & Development

### Step 1: Load the data

First of all, we will collect some data and divide them into training sets and testing sets. The major way I collected music of different bands and of different genres is via the music sharing website Free Music Archive (free music). By using the MATLAB function *webread*, we can download music based on an url. We will download eight music/songs for each band or artist, and so that we have 24 music/songs for each test. Then I choose to extract five clips from each music to form a training datasets with 120 columns (clips) in total. Notice that all the clips we use in the experiment are 5-second clips. There exist both left and right channels for each clip, and we will average the two datasets for each channel of a clip.

### Part 2: Wavelet transform - Spectrogram

As the first step of constructing a classification model, we will decompose the audio file into spectrograms. We will apply Fourier transform and use the MATLAB function *fft*. We will follow the same process as we did in the first chapter. For each clip, we set the time translation as 0.1 seconds and apply a Gaussian filter before using the Fourier transform. We will reshape the result spectrogram of frequencies into a vector and save it as a column of the music-data matrix titled with corresponding artist or band. We will get three matrices respectively representing all the spectrograms of 40 clips in the music by the corresponding artist/band.

### Part 3: SVD & LDA

Given all the spectrogram information, we are now going to design the trainer for the classifier. We will aggregate the three spectrogram-information matrices of different categories into a huge matrix! We produce an economy-size decomposition on this aggregated matrix. Then we multiply  $S$  with the transpose of  $V$  to get the projections onto principal components. And then we take 10 features of each class for later classification. Once the singular value decomposition has been performed, the linear discrimination analysis is applied immediately. We calculate the between-class  $S_B$  and within-class  $S_W$  of the featured components, just as the formula does. For our multi-class case (3 classes), we will calculate  $S_B$  by adding up the distances between the centers of each cluster and the center of the total data. Then we will use function *eig* to get the eigenvalues and eigenvectors of  $S_B$  and  $S_W$ . We extract the coefficient  $w$  by looking for the maximum ratio of the between-class scattering to the within-class scattering, and multiply the coefficients with the three feature component sets. To get the **two** thresholds, we trace each pair of computed values of two adjacent categories and get the average point of their bounds.

### Part 4: Testing

After getting two thresholds for the three classes, we then load another set of audio data to form a testing set, which incorporates music/songs from all the three bands or artists. We load the data and make spectrograms just as we did for the training set. And then we apply the PCA projection and LDA projection on the wavelet dataset to get the decision values. By comparing the decision value and the thresholds, the program predicts a classification for a music, and then we can examine the success rate for predictions of all the testing data.

## Sec IV: Computational Results & Conclusion

In test 1, I choose Ketsa (Instrumental), Lately Kind of Yeah (Rock), and Dee Yan-Key (Latin) as the three classes, and we will represent them as band1, band2, and band3. After training, I got the two thresholds:  $\text{threshold1} = 7.0377\text{e}+03$  and  $\text{threshold2} = 1.4806\text{e}+04$ . Figure 1 shows the LDA projection of the three classes: The red ones represent objects in band1; the green ones represent objects in band2; and the blue ones represent objects in band3. We get a good projection and the overall condition is pretty clear that  $\text{band2} < \text{band3} < \text{band1}$ . Figure 2 shows the statistics associated with the three classes projected onto the LDA basis and the threshold between adjacent classes, represented as red lines.

I use 45 samples as testing data and we get 10 classified incorrectly. Thus the success rate for test 1 is 0.7778. Collecting different audio data and performing the same process for test 2 and test 3, we get success rate for test 2 = 0.5111 and success rate for test 3 = 0.8000.

Thus the success rate for test 3 is the greatest, and I think that's caused by the diversity of training sets in each genre. The success rate for test 2 is the lowest, and I think it is reasonable since we choose three bands within the same genre, so that the classification becomes difficult.

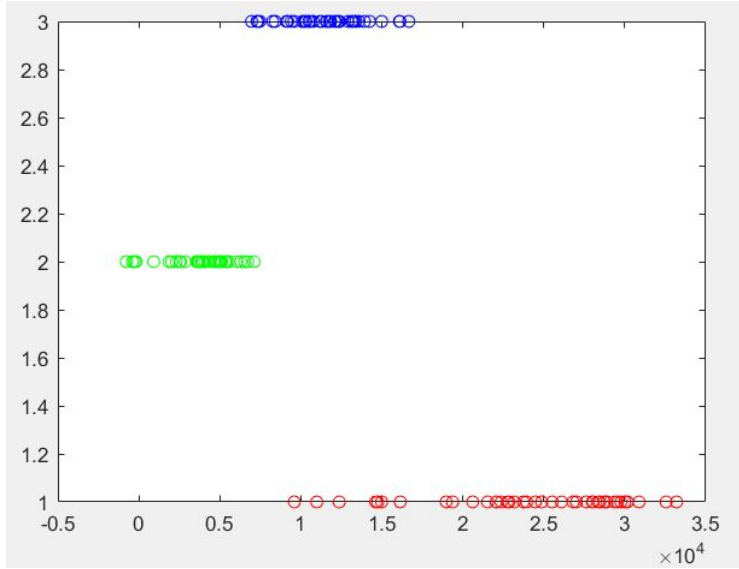


Figure 1: LDA projection of three classes

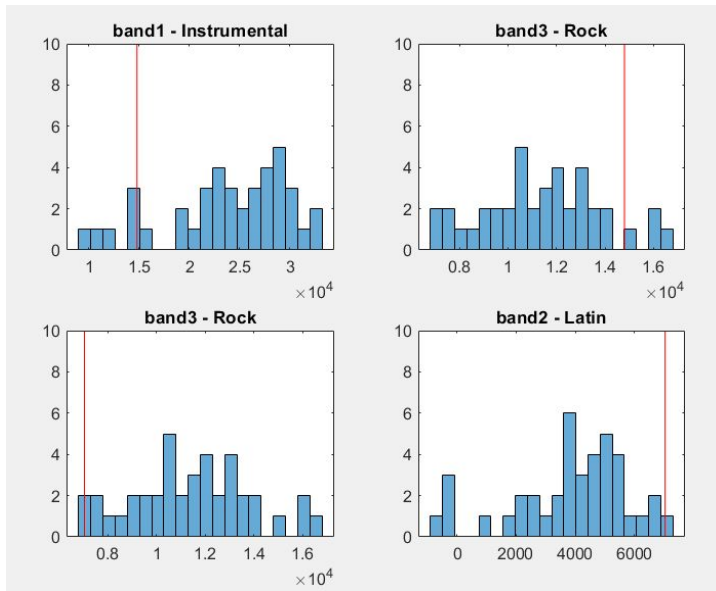


Figure 2: Histogram of the statistics associated with the three classes projected onto the LDA basis

## Appendix A: MATLAB functions used

webread: Read content from RESTful web service

svd: produces an economy-size decomposition of m-by-n matrix A.

repelem: Repeat copies of array elements.

## Appendix B: MATLAB codes

```

1 - clear; close all; clc
2
3 %%
4 % Song/Music by artist Ketsa (Instrumental)
5 - urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
6           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
7           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
8           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
9           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
10          "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
11          "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
12          "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creati
13
14 - Ketsa_clips = []; % 40 clips from the songs/music by Ketsa
15 - dur = 5; % 5 seconds clip
16 - for i = 1:length(urls)
17 -     [y, Fs] = webread(urls(i));
18 -     for j = 1:5
19 -         [y1, y2] = extract_clip2(y, Fs, 30 + j*20, dur);
20 -         y_ave = (y1 + y2) / 2;
21 -         Ketsa_clips = [Ketsa_clips y_ave];
22 -     end
23 - end
24 %%
25 - Ketsa_dat = wavelet_spectrogram(Ketsa_clips, Fs);
26
27 %% Song/Music by artist Lately Kind of Yeah (Rock)
28
29 - urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
30           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
31           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
32           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
33           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
34           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
35           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
36           "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Deco
37
38 - Lky_clips = []; % 40 clips from the songs/music by Ketsa
39 - dur = 5; % 5 seconds clip
40 - for i = 1:length(urls)
41 -     [y, Fs] = webread(urls(i));
42 -     for j = 1:5

```

```

40 - for i = 1:length(urls)
41 -     [y, Fs] = webread(urls(i));
42 -     for j = 1:5
43 -         [y1, y2] = extract_clip2(y, Fs, 30 + j*20, dur);
44 -         y_ave = (y1 + y2) / 2;
45 -         Lky_clips = [Lky_clips y_ave];
46 -     end
47 - end
48
49 %%
50 - Lky_dat = wavelet_spectrogram(Lky_clips, Fs);
51
52 %% Song/Music by artist Dee Yan-Key (Latin)
53 - urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
54 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
55 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
56 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
57 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
58 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
59 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
60 -         "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommur
61
62 - Dyk_clips = []; % 40 clips from the songs/music by Ketsa
63 - dur = 5; % 5 seconds clip
64 - for i = 1:length(urls)
65 -     [y, Fs] = webread(urls(i));
66 -     for j = 1:5
67 -         [y1, y2] = extract_clip2(y, Fs, 30 + j*30, dur);
68 -         y_ave = (y1 + y2) / 2;
69 -         Dyk_clips = [Dyk_clips y_ave];
70 -     end
71 - end
72
73 %%
74 - Dyk_dat = wavelet_spectrogram(Dyk_clips, Fs);
75
76 %%
77 - [U,S,V,w,vband1,vband2,vband3] = genre_trainer(Ketsa_dat,Lky_dat,Dyk_dat,10);
78
79 %%
80 - plot(vband1,1,'ro');hold on
81 - plot(vband2,2,'go');

```



```

80 - plot(vband1,1,'ro');hold on
81 - plot(vband2,2,'go');
82 - plot(vband3,3,'bo');
83
84 - sort1 = sort(vband1);
85 - sort2 = sort(vband2);
86 - sort3 = sort(vband3);
87
88 - threshold1 = get_threshold(sort2,sort3); % 7.0377e+03
89 - threshold2 = get_threshold(sort3,sort1); % 1.4806e+04
90 %%
91 - figure
92 - subplot(2,2,1)
93 - histogram(sort1,20); hold on, plot([threshold2 threshold2], [0 10], 'r')
94 - title('band1 - Instrumental')
95 - subplot(2,2,2)
96 - histogram(sort3,20); hold on, plot([threshold2 threshold2], [0 10], 'r')
97 - title('band3 - Rock')
98 - subplot(2,2,3)
99 - histogram(sort3,20); hold on, plot([threshold1 threshold1], [0 10], 'r')
100 - title('band3 - Rock')
101 - subplot(2,2,4)
102 - histogram(sort2,20); hold on, plot([threshold1 threshold1], [0 10], 'r')
103 - title('band2 - Latin')
104
105
106 %% Test the classifier
107 - urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Comm
108 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/
109 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/
110 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/
111 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/
112 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/
113 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommunity/Dee_Y
114 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommunity/Dee_Y
115 - "https://files.freemusicarchive.org/storage-freemusicarchive-org/music/ccCommunity/Dee_Y
116
117 - test_clips = []; % 40 clips from the songs/music by Ketsa
118 - dur = 5; % 5 seconds clip
119 - slice_time = [30 30 30 20 20 10 35 40 40];
120 - for i = 1:length(urls)

```



```

128
129 %%
130 TestNum = size(test_clips,2)
131 Test_wave = wavelet_spectrogram(test_clips, Fs); % wavelet transformation
132 TestMat = U'*Test_wave; % PCA projection
133 pval = w'*TestMat; % LDA projection
134 %%
135 hidden_labels = repelem([1 2 3], 15); % Answer to our test data
136
137 ResVec = zeros(1,TestNum); % Answer by the classifier model
138 for i = 1:TestNum
139     if pval(i) > threshold2
140         ResVec(i) = 1; % band1
141     elseif pval(i) < threshold1
142         ResVec(i) = 2; % band2
143     else
144         ResVec(i) = 3; % band3
145     end
146 end
147
148 err_num = 0;
149 for i = 1:TestNum
150     if (ResVec(i) ~= hidden_labels(i))
151         err_num = err_num + 1;
152     end
153 end
154
155 disp('Number of mistakes')
156 err_num
157 disp('Rate of success');
158 sucRate = 1-err_num/TestNum
159
160 %%
161 function musData = wavelet_spectrogram(clips_data, Fs)
162     [n,nclips] = size(clips_data);
163     L = n / Fs; % 5 secs
164     t2 = linspace(0,L,n+1);
165     t = t2(1:n);
166     tslide = 0:0.1:L;
167     musData = zeros(length(tslide) * n, nclips);
168     for i = 1:nclips % for each clip, we create a corresponding spectrogram

```

```

169 -     y = clips_data(:,i);
170 -     v = y';
171 -     % k = ((2*pi)/L)*[0:(n-1)/2 -(n-1)/2:-1]; % length of n is odd
172 -     % ks = fftshift(k);
173 -     vgt_spec = zeros(length(tslide), n);
174 -     for j = 1:length(tslide)
175 -         g = exp(-100*(t-tslide(j)).^2);
176 -         vg = g.* v;
177 -         vgt = fft(vg);
178 -         vgt_spec(j,:) = fftshift(abs(vgt));
179 -     end
180 -     musData(:, i) = reshape(vgt_spec,length(tslide)*n,1);
181 - end
182 - end
183
184
185 % Load a clip of a song with a specified url, start time and duration in
186 % seconds. Assume the input time in the range of the duration of the song.
187 ⊕ function [y1, y2, Fs] = extract_clip(url, start_time, duration) ...
203
204
205 % 2nd ver: Load a clip based on input signal.
206 ⊕ function [y1, y2] = extract_clip2(y, Fs, start_time, duration) ...
220
221
222 ⊖ function [U,S,V,w,vband1,vband2,vband3] = genre_trainer(band1,band2,band3,feature)
223 -     n1 = size(band1,2);
224 -     n2 = size(band2,2);
225 -     n3 = size(band3,2);
226 -     [U,S,V] = svd([band1 band2 band3],'econ');
227 -     genres = S*V'; % projection onto principal components
228 -     U = U(:,1:feature);
229 -     band11 = genres(1:feature,1:n1);
230 -     band22 = genres(1:feature,n1+1:n1+n2);
231 -     band33 = genres(1:feature,n1+n2+1:n1+n2+n3);
232
233 -     mband1 = mean(band11,2);
234 -     mband2 = mean(band22,2);
235 -     mband3 = mean(band33,2);
236 -     mttotal = mean([mband1 mband2 mband3],2);
237
238 -     Sw = 0; % within class variances

```

```

234 - mband2 = mean(band22,2);
235 - mband3 = mean(band33,2);
236 - mttotal = mean([mband1 mband2 mband3],2);
237
238 - Sw = 0; % within class variances
239 - for k = 1:n1
240 -     Sw = Sw + (band11(:,k)-mband1)*(band11(:,k)-mband1)';
241 - end
242 - for k = 1:n2
243 -     Sw = Sw + (band22(:,k)-mband2)*(band22(:,k)-mband2)';
244 - end
245 - for k = 1:n3
246 -     Sw = Sw + (band33(:,k)-mband3)*(band33(:,k)-mband3)';
247 - end
248 - Sb = 40*(mband1-mttotal)*(mband1-mttotal)' + 40*(mband2-mttotal)*(mband2-mttotal)'...
249 -     + 40*(mband3-mttotal)*(mband3-mttotal)';
250
251 - [V2,D] = eig(Sb,Sw); % linear discriminant analysis
252 - [~,ind] = max(abs(diag(D)));
253 - w = V2(:,ind); w = w/norm(w,2);
254
255 - vband1 = w'*band11;
256 - vband2 = w'*band22;
257 - vband3 = w'*band33;
258
259 - end
260
261
262 % method helps to determine a threshold between two adjacent categories
263 % Assume values of s1 are just lower than those of s2
264 function threshold = get_threshold(s1, s2)
265 -     t1 = length(s2);
266 -     t2 = 1;
267 -     while s2(t1) > s1(t2)
268 -         t1 = t1-1;
269 -         t2 = t2+1;
270 -     end
271 -     threshold = (s2(t1)+s1(t2))/2;
272 - end
273
274
275

```