Music Clustering and Classification

Daniel Nguyen

February 19, 2018

Abstract

Given a database of music, we apply three supervised learning techniques in order to make correct distinctions between separate genres of music. We will then use the information gathered from the learning process in order to test each algorithm's ability to classify new songs to their correct genre, and we analyze the results yielded by which algorithm to understand which algorithm works best in principle given this particular data set.

Contents

1 Introduction

Knowledge is obtained in the form of data. For humans and any other organism, that data comes in the form of senses, whether it be vision, sound, taste, etc. And in mathematics, we need raw data to start with in order to work towards building our understanding regarding any field of study. But raw data is meaningless to us if we don't have the ability to classify the given information into anything useful, relevant, or insightful. How do we find these patterns, and how do we separate these distinct patterns into groups based on various distinctions? In this paper, we will explore different techniques from the theory of data mining in order to train an algorithm to correctly distinguish between different clusters of data, and how that algorithm can be used in order to classify new pieces of data according to what it already knows from its learning. We will use the example of music to illustrate this, by using a relatively large data set of music in order to teach our algorithm which song belongs to which genre and use that information to classify songs that have not yet been encountered by the algorithm.

2 Theory

There are two types of machines learning algorithms: supervised learning, and unsupervised learning. Supervised learning is when your algorithm works based on knowledge that already exists about the data, in the sense that a person already has labels for different pieces of data in the data set and trains the algorithm to make distinctions between the data based on these labels. Unsupervised learning on the other hand does not assume anything about the data that it is given and instead tries to discover patterns based on the various attributes of the data. For this assignment we will be working mainly with supervised learning, since we already have music labeled under separate genres. We will be using Naïve Bayes, Linear Discriminant Analysis, and k-nearest neighbors in order to train and classify our songs.

2.1 Naïve Bayes

Suppose we have two classes in our data set X. We will label them with the values '0' and '1'. First, we first construct a ratio of the classes to each other, so we will take $\frac{P(1|X)}{P(2|X)}$ to be the ratio of '1''s to '0's. Bayes's Rule states that:

$$\frac{P(1|X)}{P(2|X)} = \frac{P(X|1)P(1)}{P(X|2)P(2)} \quad (1)$$

Where $P(X|1)$ is the conditional probability of X being a mistake given that it's 1, and $P(1)$ is the probability distribution of 1. What we want to do with this probability ratio is to classify a certain point according to it. For example, say we have 1000 dogs in our data set and 10 cats in our data set. Given a random point, Naïve Bayes will decide whether that point is a dog or a cat with a probabilistic model. Given a feature, it will use this feature to apply to probability equation (1) once for all classes and determine whether or not the point is a dog or cat based on which equation yields a higher score.

2.2 Linear Discriminant Analysis

The idea behind Linear Discriminant Analysis can be illustrated with the following 2-D example: Suppose we have a cluster of two classes of points lying on a plane. What the LDA does is find a line on the place such that when you project all points perpendicularly down onto that line, the maximum separation of data exists. The line is found using eigenvalues, but that is beyond the scope of this assignment. There would be a distribution of class 1 on one side and class 2 on the other, and a line of separation is drawn upwards to divide these two sides. So given a point, a decision would be made based on which side that point lies in. The LDA can be generalized to higher dimensions too, and the algorithm also has a form called Quadratic Discriminant Analysis (QDA), which tries to find not a line, but a curve that grants maximum separation of data.

2.3 K-Nearest Neighbors

Say we have two clusters of data on a plane that belong to two classes '0' and '1'. In k-nearest

neighbors, the algorithm analyzes various points on the plane and their distances to the closest data point to their location. In the case where k is 1, the algorithm would select a point and analyze the class that the data point closest to it belongs to. Any point at that location would therefore also belong to the same class, for example if the closest piece of data to a point of space is a '1', then any data point that lies in that location would also be classified as a '1'. But as we move across the plane, we might encounter a location where the closest point is instead a '0', and so any data point at that location would be a '0'. The k-nearest numbers algorithm uses these facts in order to draw a line of separation between the data set such that being on one side of the line classifies a point to the class that the cluster on that side belongs to, and classified as another class on the other side. But k can be varied; if k is 3 for instance, then it looks at its 3 nearest neighbors and classifies itself according to what the majority is classified as.

3 MATLAB Procedure

We are given a dataset of 10 genres containing 30 second samples of 100 songs for each genre. In order to cut down computational speed, all of these songs are trimmed down to 5 seconds. To get the most relevant

We have to break each of our genres into principal components by taking the SVD in order to see which features are dominant in a certain genre (since we are taking the SVD of the genre's spectrogram, we will be able to analyze the frequencies that show up often in it). In order to see how each genre projects onto our principal components, we have to turn to the matrix V.

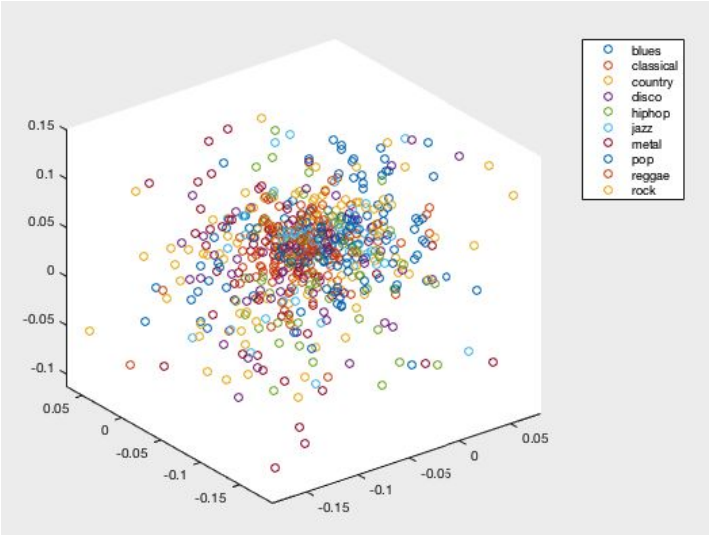

Figure 1: A 3D plot of the 1st through 3rd modes of each genre's V matrix from the SVD. This is a visualization of every song projected into 3D space.

We should take n number of modes to work with, and we can think of doing this as representing each song as a single point in n -dimensional space. With our cluster of points, we can then

apply the supervised learning algorithms to classify our songs.

Firstly, we use the randperm() command in order to generate a vector consisting of a random ordering of songs within a genre that we want, and we will use this command 10 times for each genre for good measure. The reason why we want to randomize our selection of songs is to vary our sample as much as possible to see how the algorithm performs on average when faced with a variety of arrangements in space.

Next, we build up our training set, which we will use as data on which our algorithm will come up with its particular distinctions between each genre, and our testing set, on which we will see if the algorithm manages to classify the songs in the set correctly based on the guidelines that were made in the training process. I decided to choose 70 songs from each genre for my training set, which consists of a total of 700 songs, and 30 songs from each genre from my testing set (300 songs).

We then have to label our songs. I made a 700 x 1 matrix consisting of seventy 1s stacked on top of seventy 2s and so on and so forth up to 10. This is needed information for the algorithm to make its decisions, which can most clearly be demonstrated by the k-nearest numbers algorithm, which draws separation boundaries based on the label of the nearby points it encounters.

We have all that we need to analyze the performance of our three algorithms. An important step is to cross-validate our results, so as to get a more accurate interpretation of our algorithms' performances (one might perform really well, but it could just be a probabilistic fluke. More trials will put the results of the first run to the test up to scrutiny).

To get results regarding how well our algorithms perform, we need to store the number of classifications that they get right compared to the number of classifications expected. We can then take the average of the results at every iteration and compare those averages to see how each algorithm squares up against another.

4 Results

Naïve Bayes

	1 Trial1	2 Trial2	3 Trial3	4 Trial4	5 Trial5	6 Average
1 blues	0.1333	0.6000	0.6333	0.0333	0.0667	0.2933
2 classical	0.0333	0.7000	0.0667	0	0.7333	0.3067
3 country	0.0667	0.0333	0	0.0333	0.1667	0.0600
4 disco	0.0667	0.3000	0	0.0333	0.0333	0.0867
5 hiphop	0.0333	0	0	0	0.0333	0.0133
6 jazz	0.4667	0	0.0667	0.0333	0.0333	0.1200
7 metal	0	0	0	0.4333	0.0333	0.0933
8 pop	0.0333	0.0333	0.0333	0	0.4000	0.1000
9 reggae	0.5000	0	0.1667	0.0333	0.1333	0.1667
10 rock	0.0667	0.0333	0	0.1333	0.0333	0.0533

Figure 1: Accuracy results for the Naïve Bayes algorithm

Linear Discriminant Analysis

	1 Trial1	2 Trial2	3 Trial3	4 Trial4	5 Trial5	6 Average
1 blues	0.3000	0.3333	0.3000	0.3000	0.2000	0.2867
2 classical	0.7333	0.6333	0.7333	0.5667	0.6667	0.6667
3 country	0.1333	0.0667	0.1333	0.2000	0.2000	0.1467
4 disco	0.0667	0.1333	0	0.0333	0.0667	0.0600
5 hiphop	0.0333	0.0333	0	0.0333	0.0333	0.0267
6 jazz	0.1000	0.0333	0.0333	0.0333	0.0333	0.0467
7 metal	0.1000	0.0333	0	0.1333	0.3333	0.1200
8 pop	0	0.0667	0.1000	0.0333	0.0333	0.0467
9 reggae	0.1333	0.2000	0.1333	0.0333	0.1333	0.1267
10 rock	0	0.1000	0	0	0.0333	0.0267

Figure 2: Accuracy results for the LDA algorithm

K-Nearest Neighbors

	1 Trial1	2 Trial2	3 Trial3	4 Trial4	5 Trial5	6 Average
1 blues	0.7333	0.7333	0.5333	0.5333	0.5667	0.6200
2 classical	0.7000	0.6000	0.6667	0.6667	0.6000	0.6467
3 country	0.4000	0.2333	0.4000	0.2000	0.2333	0.2933
4 disco	0.2333	0.3333	0.3000	0.4000	0.2667	0.3067
5 hiphop	0.2667	0.3667	0.1333	0.3333	0.3333	0.2867
6 jazz	0.5667	0.3000	0.4333	0.5000	0.6000	0.4800
7 metal	0.3667	0.4333	0.5667	0.5000	0.5000	0.4733
8 pop	0.3667	0.3333	0.4667	0.5333	0.3667	0.4133
9 reggae	0.4667	0.5333	0.4667	0.6000	0.5333	0.5200
10 rock	0.2333	0.4000	0.4000	0.3333	0.5000	0.3733

Figure 3: Accuracy results for the K-nearest neighbors algorithm

From the data, we can see that the k-nearest neighbors yielded the highest accuracy rates, with a range roughly in between 30%-65%. In contrast, the Naïve Bayes algorithm performed the worst of the three, with many genres having an accuracy less than 10%! We can see that LDA does slightly better than Naïve Bayes, with its performance for classical averaging to 66%.

We can see why Naïve Bayes has such a bad performance in this data set. If we refer back to Figure 1, a great majority of the points were concentrated and mixed into a condensed area. Also, there were an equal number of songs for each genre, giving any data point an unvarying 10% chance of being a certain genre.

LDA also is not the most optimal algorithm to use in a case like this. When running our analysis, we can clearly get a sense of which algorithm works best for our task. Our data is not separated

into nice clusters and there is a lot of mixing, as can be seen in Figure 1, so the task finding a region to project the points into with the aim of extracting 10 distributions of data is difficult, and will not yield accurate boundaries to serve as litmus tests.

K-nearest neighbors proved to be the best algorithm for our data. It analyzed the n-dimensional space given and took every single point's location in consideration in order to form very fluid dividing regions. As shown in Figure 4, we can see a rough staircase pattern, which is very good as a staircase pattern in a bar plot is what a 100% accurate prediction algorithm would yield.

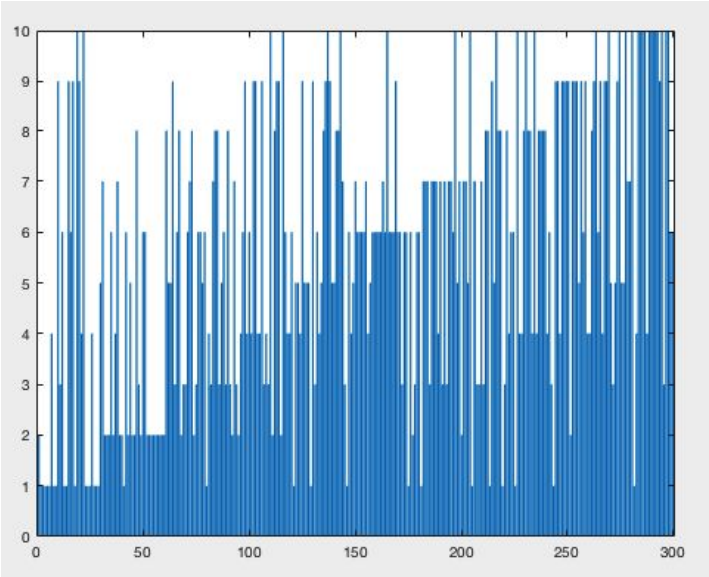

Figure 4: The bar plot of the predictions for k-nearest neighbors.

None of the techniques that we have applied have broken into the 80%-90% range for accuracy though. We can attribute this to the songs' length being only 5 seconds, and those 5 seconds are likely to be similar to the other songs' 5 seconds, therefore not providing enough data to be analyzed. These were just three techniques, and further investigation can be made to more data mining techniques in order to find improvements.

5 Summary and Conclusions

Classification of data can be done through several methods; by finding patterns in data through unsupervised learning and through analyzing the features of data in supervised learning. We explored supervised learning algorithms in this paper, which were Naïve Bayes, Linear Discriminant Analysis, and K-nearest neighbors. According to our results, we found that k-nearest neighbors algorithm was by far the superior algorithm in comparison with the other two algorithms. We also concluded that our algorithms if our data set had larger samples, our accuracy range would be higher.

Appendix A MATLAB Functions Used

audioread();
spectrogram();
hamming();
svd();
randperm();
fitcnb(); %naive bayes
classify(); %LDA
knnsearch(); %k-nearest neighbors
bar();
table();

Appendix B MATLAB Code

```
clear all;
close all;
clc;

cd '/Users/daniel/MATLAB/Classifier/Music'

musicfiles = dir(pwd);
songname = dir('*wav');

songs = [];
sr = 44100; % sampling rate obtained from sampling first few songs
for i = 3:length(musicfiles)
    [Y, FS] = audioread(musicfiles(i).name, [7*sr, 12*sr]);
    signal = Y(:,1) + Y(:,2);
    songs(:,i-2) = signal;
end

%%

sgrams = zeros(146718,1000);
window = hamming(1024);
noverlap = 256;
for i = 1:length(songs(1,:))
    S = spectrogram(songs(:,i), window, noverlap);
    S = abs(S);
    spsize = size(S);
    S = reshape(S, [spsize(1) * spsize(2), 1]);
    sgrams(:,i) = S;
end

%%

blues = sgrams(:,1:100);
classical = sgrams(:,101:200);
country = sgrams(:,201:300);
disco = sgrams(:,301:400);
hiphop = sgrams(:,401:500);
jazz = sgrams(:,501:600);
metal = sgrams(:,601:700);
pop = sgrams(:,701:800);
reggae = sgrams(:,801:900);
rock = sgrams(:,901:1000);

%%

[U1, S1, V1] = svd(blues,'econ');
[U2, S2, V2] = svd(classical,'econ');
[U3, S3, V3] = svd(country,'econ');
[U4, S4, V4] = svd(disco,'econ');
[U5, S5, V5] = svd(hiphop,'econ');
[U6, S6, V6] = svd(jazz,'econ');
```

```
[U7, S7, V7] = svd(metal,'econ');
[U8, S8, V8] = svd(pop,'econ');
[U9, S9, V9] = svd(reggae,'econ');
[U10, S10, V10] = svd(rock,'econ');

%%
figure(1)
subplot(5,2,1)
plot(diag(S1),'ko')
subplot(5,2,2)
plot(diag(S2),'ko')
subplot(5,2,3)
plot(diag(S3),'ko')
subplot(5,2,4)
plot(diag(S4),'ko')
subplot(5,2,5)
plot(diag(S5),'ko')
subplot(5,2,6)
plot(diag(S6),'ko')
subplot(5,2,7)
plot(diag(S7),'ko')
subplot(5,2,8)
plot(diag(S8),'ko')
subplot(5,2,9)
plot(diag(S9),'ko')
subplot(5,2,10)
plot(diag(S10),'ko')

%%
mode_low = 1;
mode_high = 10;

%%

figure(2)
plot3(V1(:,mode_low),V1(:,mode_low+1),V1(:,mode_low+2),'o')
hold on
plot3(V2(:,mode_low),V2(:,mode_low+1),V2(:,mode_low+2),'o')
hold on
plot3(V3(:,mode_low),V3(:,mode_low+1),V3(:,mode_low+2),'o')
hold on
plot3(V4(:,mode_low),V4(:,mode_low+1),V4(:,mode_low+2),'o')
hold on
plot3(V5(:,mode_low),V5(:,mode_low+1),V5(:,mode_low+2),'o')
hold on
plot3(V6(:,mode_low),V6(:,mode_low+1),V6(:,mode_low+2),'o')
hold on
plot3(V7(:,mode_low),V7(:,mode_low+1),V7(:,mode_low+2),'o')
hold on
plot3(V8(:,mode_low),V8(:,mode_low+1),V8(:,mode_low+2),'o')
hold on
plot3(V9(:,mode_low),V9(:,mode_low+1),V9(:,mode_low+2),'o')
hold on
plot3(V10(:,mode_low),V10(:,mode_low+1),V10(:,mode_low+2),'o')
```

```
legend('blues','classical','country','disco','hiphop','jazz','metal','pop','re
ggae','rock');

%% Training with Cross-Validation

totalscores1 = [];
totalscores2 = [];
totalscores3 = [];


c_v_iter = 5;

for i = 1:c_v_iter

q1 = randperm(100);
q2 = randperm(100);
q3 = randperm(100);
q4 = randperm(100);
q5 = randperm(100);
q6 = randperm(100);
q7 = randperm(100);
q8 = randperm(100);
q9 = randperm(100);
q10 = randperm(100);

xblues = V1(:,mode_low:mode_high);
xclassical = V2(:,mode_low:mode_high);
xcountry = V3(:,mode_low:mode_high);
xdisco = V4(:,mode_low:mode_high);
xhiphop = V5(:,mode_low:mode_high);
xjazz = V6(:,mode_low:mode_high);
xmetal = V7(:,mode_low:mode_high);
xpop = V8(:,mode_low:mode_high);
xreggae = V9(:,mode_low:mode_high);
xrock = V10(:,mode_low:mode_high);

xtrain = [xblues(q1(1:70),:)
    xclassical(q2(1:70),:)
    xcountry(q3(1:70),:)
    xdisco(q4(1:70),:)
    xhiphop(q5(1:70),:)
    xjazz(q6(1:70),:)
    xmetal(q7(1:70),:)
    xpop(q8(1:70),:)
    xreggae(q9(1:70),:)
    xrock(q10(1:70),:)];

xtest = [xblues(q1(71:100),:)
    xclassical(q2(71:100),:)
    xcountry(q3(71:100),:)
    xdisco(q4(71:100),:)
    xhiphop(q5(71:100),:)
    xjazz(q6(71:100),:)
    xmetal(q7(71:100),:)
```

```
    xpop(q8(71:100),:)
    xreggae(q9(71:100),:)
    xrock(q10(71:100),:)];

%%

ctrain = [ones(70,1)
    2*ones(70,1)
    3*ones(70,1)
    4*ones(70,1)
    5*ones(70,1)
    6*ones(70,1)
    7*ones(70,1)
    8*ones(70,1)
    9*ones(70,1)
    10*ones(70,1)];

%%

ctrain3 = [ones(70,3)
    2*ones(70,3)
    3*ones(70,3)
    4*ones(70,3)
    5*ones(70,3)
    6*ones(70,3)
    7*ones(70,3)
    8*ones(70,3)
    9*ones(70,3)
    10*ones(70,3)];

%% Naive Bayes

nb = fitcnb(xtrain,ctrain);
pre1 = nb.predict(xtest);

bar(pre1)
score = 0;
scores1 = zeros(1,10);
for i = 1:length(pre1)
    if pre1(i) == (floor(i/30) + 1)
        score = score + 1;
    end
    if i~= 1 && (mod(i,30) == 1)
        scores1(floor(i/30)) = score / 30.0;
        score = 0;
    end
    if i == length(pre1)
        scores1(floor(i/30)) = score / 30.0;
    end
end

%% Linear Discriminant Analysis

pre2 = classify(xtest,xtrain,ctrain);
```

```
bar(pre2)
score = 0;
scores2 = zeros(1,10);
for i = 1:length(pre2)
    if pre2(i) == (floor(i/30) + 1)
        score = score + 1;
    end
    if i~= 1 && (mod(i,30) == 1)
        scores2(floor(i/30)) = score / 30.0;
        score = 0;
    end
    if i == length(pre2)
        scores2(floor(i/30)) = score / 30.0;
    end
end

%% k-nearest neighbors

[IDX, D] = knnsearch(xtrain,xtest);
pre3=[];
for i=1:length(xtest(:,1))
    pre3(i) = ctrain(IDX(i));
end
bar(pre3);
scores3 = zeros(1,10);
for i = 1:length(pre3)
    if pre3(i) == (floor(i/30) + 1)
        score = score + 1;
    end
    if i~= 1 && (mod(i,30) == 1)
        scores3(floor(i/30)) = score / 30.0;
        score = 0;
    end
    if i == length(pre3)
        scores3(floor(i/30)) = score / 30.0;
    end
end

totalscores1 = [totalscores1; scores1];
totalscores2 = [totalscores2; scores2];
totalscores3 = [totalscores3; scores3];

end

averagescores1 = [];
averagescores2 = [];
averagescores3 = [];
for i = 1:length(scores1)
    averagescores1(:,i) = sum(totalscores1(:,i)) / c_v_iter;
    averagescores2(:,i) = sum(totalscores2(:,i)) / c_v_iter;
    averagescores3(:,i) = sum(totalscores3(:,i)) / c_v_iter;
end

%%
```

```
genrenames =
{'blues','classical','country','disco','hiphop','jazz','metal','pop','reggae',
'rock'};

tablem1 = [totalscores1; averagescores1].';
tablem2 = [totalscores2; averagescores2].';
tablem3 = [totalscores3; averagescores3].';

table1 = table(tablem1(:,1), tablem1(:,2), tablem1(:,3), tablem1(:,4),
tablem1(:,5), tablem1(:,6));
table2 = table(tablem2(:,1), tablem2(:,2), tablem2(:,3), tablem2(:,4),
tablem2(:,5), tablem2(:,6));
table3 = table(tablem3(:,1), tablem3(:,2), tablem3(:,3), tablem3(:,4),
tablem3(:,5), tablem3(:,6));

table1.Properties.VariableNames = {'Trial1' 'Trial2' 'Trial3' 'Trial4'
'Trial5' 'Average'};
table2.Properties.VariableNames = {'Trial1' 'Trial2' 'Trial3' 'Trial4'
'Trial5' 'Average'};
table3.Properties.VariableNames = {'Trial1' 'Trial2' 'Trial3' 'Trial4'
'Trial5' 'Average'};
table1.Properties.RowNames = genrenames;
table2.Properties.RowNames = genrenames;
table3.Properties.RowNames = genrenames;
```