Music Genre Identification

Chongyi Xu
University of Washington
AMATH 482/582 Winter Quarter 2018
chongyix@uw.edu

March 10, 2018

Abstract

For this project, we are focusing on using machine learning techniques to classify different music bands/-genres. There are generally two techniques to be used in this project - Naive Bayes, and Decision Trees.

1 Introduction and Overview

1.1 Introudction

Humans classify audio signals all the time without paying attentions. Classifying who is speaking in a group, or realizing the person on the other side of the phone call, or classifying whether the music that is playing on the radio is rock or jazz. But the problem comes that how could we tell machine to know that, who is speaking (cocktail party problem), or what genre does the playing music belong to.

1.2 Overview

The project has three tests to be completed.

- Band Classification
- The Case for Seattle
- Genre Classification

I will pick 6 songs for each of Michael Jackson, Soundgarden, and Beethoven for the Band Classification Test. For the Seattle Case Test, I will pick Soundgarden, Alice in Chains, and Pearl Jam to classify 5-sound sound clips with also 6 songs for each artists. And for the general Genre Classification, I will use the data set from Marsyas.
Marsyas has approximately 1.2GB of different genres of music. This data set was used for the well known paper in genre classification " Musical genre classification of audio signals " by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002. And it consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in .wav format. I will choose three genres out of the given 10 genres (rock, jazz, classical).

2 Theoretical Background

The machine learning skills I will use for the classification are the most-known classifiers. Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with naive independence assumptions between the features. Naive Bayes classifiers are highly salable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.
Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree

structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

3 Algorithm Implementation and Development

3.1 General Procedure

- Load the music files. For every band or genre of music, reshape the five-second-audio-clip vector to a data matrix that each column contains a five-second clip. And to keep the data sizes manageable, it is necessary to resample the data matrix.
- Label the clips to keep tracking of which column belongs to who/what genre.
- Collect representative features for each artist/-genre such that we could distinguish one from others. The representative features includes
  - How many times the signal goes across zero
  - The maximum, and minimum absolute value of the signal
  - The mean value, variance, first norm value of the signal
  - The largest frequencies that appears in the signal.
- After collecting the data, construct a matrix contains all features information and divide the matrix into training set and testing set. The way I choose my training/testing sets is that randomly pick 70% of the data to be training data and the other 30% to be testing data.
- Then perform SVD on the data matrix. Pick the modes with more than 10% energy such that it will have a better performance on prediction.
- Train the data set using different classifiers (in this case, decision tree algorithm and naive Bayes classifier).
- Test the prediction and collect the accuracy data.

In order to have a reasonable result, I will train the data and get the accuracy result for 5 times and take the average value.

Table 1: Accuracy for Test 1

	Naive Bayes	**Decision Tree**
Trail 1	66.5823	75.9494
Trail 2	69.1139	79.7468
Trail 3	70.3797	78.2278
Trail 4	66.8354	80.2532
Trail 5	71.3924	78.9873
Average	68.8608$\pm$ 4.5185	78.6329$\pm$ 2.8393

Table 2: Accuracy for Test 2

	Naive Bayes	**Decision Tree**
Trail 1	56.6092	67.2414
Trail 2	55.1724	72.7011
Trail 3	56.8966	72.4138
Trail 4	53.1609	73.8506
Trail 5	56.3218	74.7126
Average	55.6322$\pm$ 2.3368	72.1839$\pm$ 8.4803

4 Computational Results

4.1 Test 1

From Table 1, we can see that the overall result is pretty impressive. $68.86 \pm 4.5\%$ for naive Bayes classifier and $78.63 \pm 3.8\%$ for decision tree algorithm. And the point is that this result is only from 6 sample songs. It is reasonable to predict that with more training data, these two algorithms might give a even better performance.

4.2 Test 2

From Table 2, obviously that the overall accuracy has dropped for a little but still acceptable. Using decision tree would have approximately 72.18% accuracy during the testing. And test 2 generally results in a lower accuracy is also reasonable since the three bands (Alice in Chains, Soundgarden and Pearl Jam) are in the same genre. Their music styles are close which might even affect human's decision.

Table 3: Accuracy for Test 2

	Naive Bayes	**Decision Tree**
Trail 1	50.7407	55.3704
Trail 2	52.7778	50.9259
Trail 3	52.4074	54.4444
Trail 4	55.3704	51.2963
Trail 5	50.9259	55.3704
Average	52.4444± 3.4705	53.4815± 4.8422

4.3 Test 3

According to Table 3, the accuracy in test 3 is not as ideal as the previous ones. Some problem to the classifiers might be

- The genre itself has be developing over time. ⇒ For instance, the rock music in 70s should be quite different from rock music recently.
- The decision tree does not have enough branches to have ideal performance. ⇒ This happens due to the limitation of my personal laptop. If I choose to have a larger number of branches for the decision tree, the run time will be extremely costly.

5 Summary and Conclusions

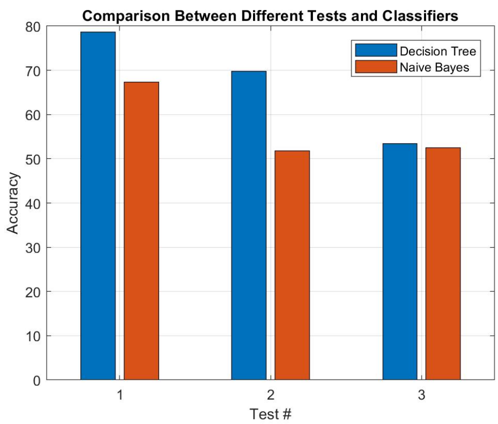

Figure 1: Accuracy Comparison

From the graph Figure 1, we can see that generally, decision tree algorithm has a better overall result comparing to naive Bayes classifier. This is reasonable since that we could consider how iTunes Store divides all the artists into categories. And we will use decision tree processes to look for the certain artist.
And for the 3-rd test - genre classification, decision tree algorithm and naive Bayes classifier have kind of the same performance, which is also understandable since when we distinguish one genre from another, we will use not only categories but also some "true-false" to classify genre.

```matlab
%% TEST 1
clc; clear all; close all;
%% Import MP3 Files
% Choice: Michael Jackson, Soundgarden, Beethoven
% myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
% [mj, Rmj] = importData([myDir '\MichaelJackson']);
% [sg, Rsg] = importData([myDir '\SoundGarden']);
% [bee, Rbee] = importData([myDir '\Beethoven']);
%
% % Resample the data
% mj = resample(mj, 1, 2);
% sg = resample(sg, 1, 2);
% bee = resample(bee, 1, 2);
%
% % Save for further use
% save('mj.mat', 'mj');
% save('sg.mat', 'sg');
% save('bee.mat', 'bee');

% Load if possible
load('mj.mat');
load('sg.mat');
load('bee.mat');

% Make into one data matrix
X = [mj, sg, bee];

%% Label

for i = 1:size(mj, 2)
    labs{i} = 'MichaelJackson';
end

for i = 1:size(sg, 2)
    labs{i+size(mj, 2)} = 'SoundGarden';
end

for i = 1:size(bee,2)
    labs{i+size(mj, 2)+size(sg, 2)} = 'Beethoven';
end

%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));

    % Get the first 2 highest frequencies from the FFT.
    fs = abs(fft(X(:,i)));
    fs = fs(1:floor(end/2));
    [~,f1(i,:)] = max(fs);
    fs(f1(i)) = 0;
    [~,f2(i,:)] = max(fs);
end

% Create the feature matrix
```

```matlab
featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2];

%% Train the classifier and get the test result for 5 loops
acc_dt = zeros(5, 1);
acc_nb = zeros(5, 1);
for i = 1:5
    acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
    acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
end

%% TEST 2
clc; clear all; close all;
%% Import MP3 Files
% % Choice: Alice In Chains, Soundgarden, Pearl Jam
% myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\MP3';
% [aic, Raic] = importData([myDir '\AliceInChains']);
% [sg, Rsg] = importData([myDir '\SoundGarden']);
% [pj, Rpj] = importData([myDir '\PearlJam']);
%
% % Resample the data
% aic = resample(aic, 1, 2);
% sg = resample(sg, 1, 2);
% pj = resample(pj, 1, 2);
%
% % Save for further use
% save('aic.mat', 'aic');
% save('sg.mat', 'sg');
% save('pj.mat', 'pj');

% Load if possible
load('aic.mat');
load('sg.mat');
load('pj.mat');

% Make into one data matrix
X = [aic, sg, pj];

%% Label

for i = 1:size(aic, 2)
    labs{i} = 'AliceInChains';
end

for i = 1:size(sg, 2)
    labs{i+size(aic, 2)} = 'SoundGarden';
end

for i = 1:size(pj,2)
    labs{i+size(aic, 2)+size(sg, 2)} = 'PearlJam';
end

%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));
```

```matlab

    % Get the first 2 highest frequencies from the FFT.
    fs = abs(fft(X(:,i)));
    fs = fs(1:floor(end/2));
    [~,f1(i,:)] = max(fs);
    fs(f1(i)) = 0;
    [~,f2(i,:)] = max(fs);

    wavl(i,:) = sum(abs(diff(X(:,i))));
end

% Create the feature matrix
featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2, wavl];

%% Train the classifier and get the test result for 5 loops
acc_dt = zeros(5, 1);
acc_nb = zeros(5, 1);
for i = 1:5
    acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
    acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
end

%% TEST 3
clc; clear all; close all;
%% Import MP3 Files
% Choice: Alice In Chains, Soundgarden, Pearl Jam
myDir = 'C:\Users\Johnnia\Desktop\46\UW\SchoolWorks\2018Winter\AMATH482\HW3\genres';
[blues, Rb] = importData([myDir '\blues']);
[classical, Rclas] = importData([myDir '\classical']);
[country, Rcoun] = importData([myDir '\country']);
[disco, Rd] = importData([myDir '\disco']);
[hiphop, Rhh] = importData([myDir '\hiphop']);
[jazz, Rjz] = importData([myDir '\jazz']);
[metal, Rm] = importData([myDir '\metal']);
[pop, Rp] = importData([myDir '\pop']);
[reggae, Rr] = importData([myDir '\reggae']);
[rock, Rrc] = importData([myDir '\rock']);

%% Resample the data
blues = resample(blues, 1, 2);
classical = resample(classical, 1, 2);
country = resample(country, 1, 2);
disco = resample(disco, 1, 2);
hiphop = resample(hiphop, 1, 2);
hiphop = hiphop(:, 1:end-1);
jazz = resample(jazz, 1, 2);
metal = resample(metal, 1, 2);
pop = resample(pop, 1, 2);
reggae = resample(reggae, 1, 2);
rock = resample(rock, 1, 2);

% Save for further use
save('blues.mat', 'blues');
save('classical.mat', 'classical');
save('country.mat', 'country');
save('disco.mat', 'disco');
save('hiphop.mat', 'hiphop');
save('jazz.mat', 'jazz');
save('metal.mat', 'metal');
save('pop.mat', 'pop');
save('reggae.mat', 'reggae');
save('rock.mat', 'rock');

%% Load if possible
load('blues.mat');
```

```
load('classical.mat');
load('country.mat');
load('disco.mat');
load('hiphop.mat');
load('jazz.mat');
load('metal.mat');
load('pop.mat');
load('reggae.mat');
load('rock.mat');

% Make into one data matrix
X = [blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock];

%% Label

for i = 1:size(blues, 2)
    labs{i} = 'blues';
end

for i = 1:size(classical, 2)
    labs{i+size(blues, 2)} = 'classical';
end

for i = 1:size(country,2)
    labs{i+size(blues, 2)+size(classical, 2)} = 'country';
end

for i = 1:size(disco,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)} = 'disco';
end

for i = 1:size(hiphop,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)} = 'hiphop';
end

for i = 1:size(jazz,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)} = 'jazz';
end

for i = 1:size(metal,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)} = 'metal';
end

for i = 1:size(pop,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)} = 'pop';
end

for i = 1:size(reggae,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)+size(pop, 2)} = 'reggae';
end

for i = 1:size(rock,2)
    labs{i+size(blues, 2)+size(classical, 2)+size(country, 2)...
        +size(disco, 2)+size(hiphop, 2)+size(jazz, 2)...
        +size(metal, 2)+size(pop, 2)+size(reggae, 2)} = 'rock';
end

%% Label2
```

```
for i = 1:size(rock, 2)
    labs{i} = 'rock';
end

for i = 1:size(jazz, 2)
    labs{i+size(rock, 2)} = 'jazz';
end

for i = 1:size(classical,2)
    labs{i+size(rock, 2)+size(jazz, 2)} = 'classical';
end
%% Get Distinguishable Features

zcd = dsp.ZeroCrossingDetector;
for i = 1:size(X,2)

    % Count the number of times the signal crosses zero.
    cross0(i,:) = double(zcd(X(:,i)));

    % Get the max, min, mean, variance and norm of the signal
    maxV(i,:) = max(abs(X(:,i)));
    minV(i,:) = min(abs(X(:,i)));
    meanV(i,:) = mean(X(:,i));
    varV(i,:) = var(X(:,i));
    normV(i,:) = norm(X(:,i));

    % Get the first 2 highest frequencies from the FFT.
    fs = abs(fft(X(:,i)));
    fs = fs(1:floor(end/2));
    [~,f1(i,:)] = max(fs);
    fs(f1(i)) = 0;
    [~,f2(i,:)] = max(fs);

    wavl(i,:) = sum(abs(diff(X(:,i))));
end

% Create the feature matrix
featureData = [cross0, maxV, minV, meanV, varV, normV, f1, f2, wavl];

%% Train the classifier and get the test result for 5 loops
acc_dt = zeros(5, 1);
acc_nb = zeros(5, 1);
for i = 1:5
    acc_dt(i) = Classifier(featureData, labs, 'DecisionTree');
    acc_nb(i) = Classifier(featureData, labs, 'NaiveBayes');
end


%%
clc; clear all; close all;
%% Comparison
load('dt_1.mat'); load('nb_1.mat');
dt1 = acc_dt; nb1 = acc_nb;
load('dt_2.mat'); load('nb_2.mat');
dt2 = acc_dt; nb2 = acc_nb;
load('dt_3.mat'); load('nb_3.mat');
dt3 = acc_dt; nb3 = acc_nb;

dt1_mean = mean(dt1); dt1_var = var(dt1);
nb1_mean = mean(nb1); nb1_var = var(nb1);
dt2_mean = mean(dt2); dt2_var = var(dt2);
nb2_mean = mean(nb2); nb2_var = var(nb2);
dt3_mean = mean(dt3); dt3_var = var(dt3);
nb3_mean = mean(nb3); nb3_var = var(nb3);
```

```
dtmeans = [dt1_mean, dt2_mean, dt3_mean];
nbmeans = [nb1_mean, nb2_mean, nb3_mean];
dtvars = [dt1_var, dt2_var, dt3_var];
nbvars = [nb1_var, nb2_var, nb3_var];

figure
hb = bar([1, 2, 3], [dtmeans, nbmeans]);
set(hb(1), 'FaceColor','r')
set(hb(2), 'FaceColor','b')
set(hb(3), 'FaceColor','g')

%%
data = [[dt1_mean, nb1_mean]; [dt2_mean, nb2_mean]; [dt3_mean, nb3_mean]];
figure(1)
b = bar(data);
err = [[dt1_var, nb1_var]; [dt2_var, nb2_var]; [dt3_var, nb3_var]];
hold on;
title('Comparison Between Different Tests and Classifiers')
xlabel('Test #');
ylabel('Accuracy');
legend('Decision Tree', 'Naive Bayes');
grid on;

function accuracy = Classifier(dat,labs, classifier)
n = size(dat,1);
trainSize = floor(n*0.7);
testSize = floor(n*0.3);

%% Divide data to be training set and testing set
randIndex = randperm(n);
randData = dat(randIndex,:);
randlabs = labs(randIndex);

% First 70% to be training data
train = randData(1:trainSize,:);
trainlabs = randlabs(1:trainSize);

% The other 30% to be testing data
test = randData(trainSize+1:end,:);
testlabs = randlabs(trainSize+1:end);

%% Perform SVD

[u,s,v] = svd(train','econ');

% Compute energy
energy = diag(s)/sum(diag(s));

% Find modes with less than 10% energy
i = find(energy < 0.1);

% If the above all modes are within 10%
if isempty(i)
    endInd = size(v,2);
else
    endInd = i(1)-1;
end

% Get the data with only modes with more than 10% energy
train = v(:,1:endInd);
test = s\u'*test';
test = test(1:endInd,:)';

%% Train
if (strcmp(classifier, 'DecisionTree'))
```

```
    model = TreeBagger(30, train, trainlabs);
elseif (strcmp(classifier, 'NaiveBayes'))
    model = fitcnb(train, trainlabs);
end
%% Test

predictions{testSize,1} = [];
correctness = zeros(testSize, 1);

for i = 1:testSize
    predictions{i,:} = predict(model, test(i,:));
    correctness(i) = strcmp(predictions{i,:}, testlabs{i});
end

accuracy = 100*sum(correctness)/length(correctness);

end

function [dat, Fs] = importData(myDir)

musics = dir(fullfile(myDir));
musics = musics(3:end, :);
n = length(musics);

dat = [];

for i = 1:n
    [song, Fs] = audioread(fullfile(myDir, musics(i).name));
    vectorSong = song(:,1);
    dat = [dat; vectorSong];
end

% Get the number of frames in 5 seconds
nIn5 = Fs*5;

% Get the number of 5-second clips
nClips = floor(length(dat)/nIn5);

% Trim the data matrix
dat = dat(1:nIn5*nClips);

% Reshape the data matrix
dat = reshape(dat,[nIn5, nClips]);

end
```