Homework 4: Music Classification

Alyssa Jin

03/06/2020

Abstract

In this assignment, I need to write a code that can classify a given piece of music by sampling a 5 second clip. I use Singular Value Decomposition (SVD) to spectrogram of songs versus the songs themselves. I re-sample the data in order to keep the data sizes more manageable.Then, I can use the characteristics of different genres of music to identify the classification of music.

I. Introduction and Overview

We can easily tell the genres of different music, whether it be K-POP, Trot, Chinese Rock, Beijing Opera, etc. The brain can always classifies the information and makes a decision based on hearing a new piece of music. I will do three tests in this assignment. Test 1 is "Band Classification", which is to consider three different bands of different genres. I pick G-DRAGON (K-POP), Hong Jin Young (Trot), Wang Feng (Chinese Rock). I take 5-second clips from a variety of each of their music, build training set, and see if I can build a statistical testing algorithm capable of accurately identifying new 5-second clips of music, then report the accuracy. Test 2 is "The Case for Seattle": repeating test 1, but with three bands from within the same genre, which is k-pop. I pick G-DRAGON, Wanna One, Turbo. Test 3 is "Genre Classification", which is to use algorithms above to broadly classify songs. I choose K-POP, Trot, Chinese Rock, and build a classifier. I choose G-DRAGON, Wanna One, Turbo for K-POP; Hong Jin Young, Park Hyun Bin, Lee Sun Hee for Trot; Wang Feng, Cui Jian, Luo Qi for Chinese Rock.

II. Theoretical Background

II.i. Singular Value Decomposition (SVD)

The vector $\mathbf{x}$ when multiplied by a matrix $\mathbf{A}$ produces a new vector $\mathbf{y}$ is aligned in a new direction with a new length. Generically, matrix multiplication will rotate and stretch (compress) a given vector as prescribed by the matrix $\mathbf{A}$. A singular value decomposition (SVD) is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications. In compact matrix notation, $\mathbf{AV} = \widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}$. Therefore the SVD of $\mathbf{A}$ is $\widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}\mathbf{V}^*$: $\mathbf{U} \in \mathbb{C}^{m\times m}$ is unitary, $\mathbf{V} \in \mathbb{C}^{n\times n}$ is unitary, $\mathbf{\Sigma} \in \mathbb{C}^{m\times n}$ is diagonal with non-negative elements and ordered from largest to smallest from element in left-top to element in right-bottom.

Theorem: Every matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ has a singular value decomposition. Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined, and, if $\mathbf{A}$ is square and the $\{\sigma_j\}$ distinct, the singular vectors $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ are uniquely determined up to complex signs.

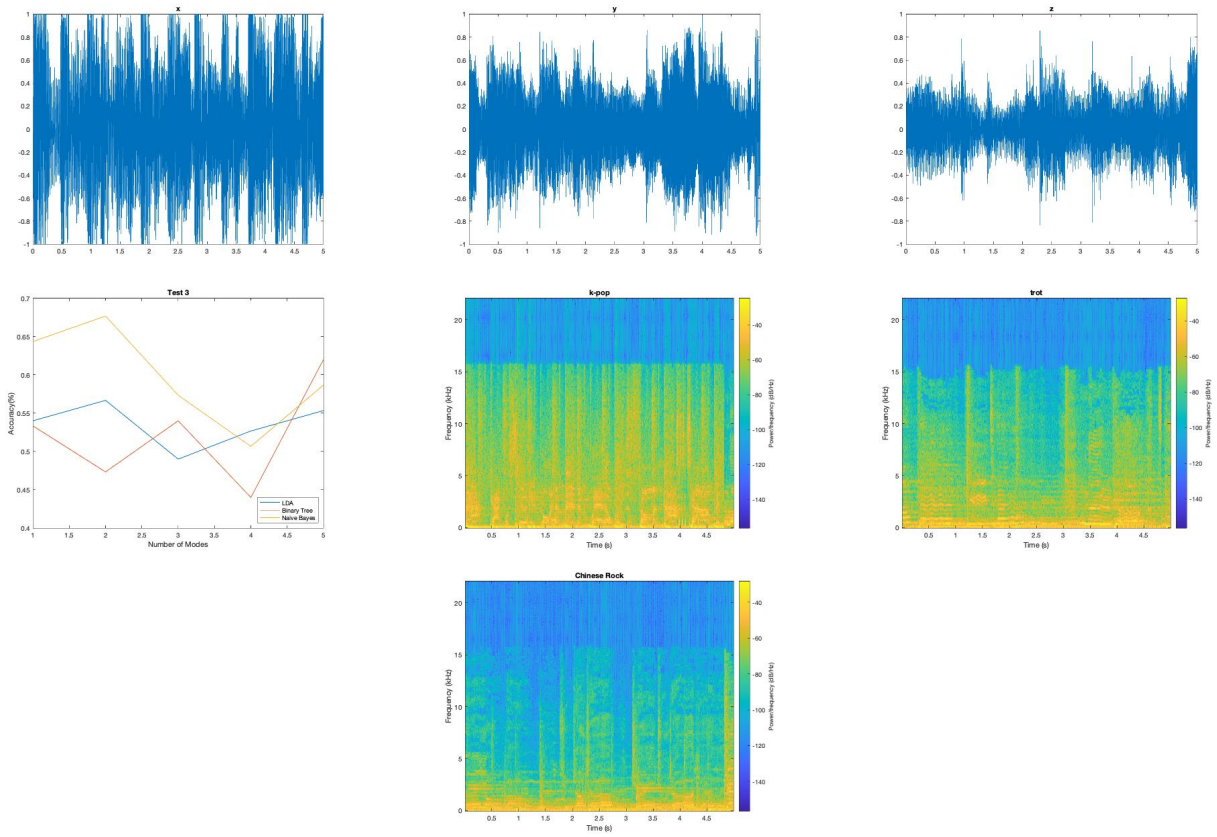
II.ii. Linear discriminant analysis (LDA)
Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.
LDA is closely related to principal component analysis (PCA) because they both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA, in contrast, does not take into account any difference in class.
LDA works when the measurements made on independent variables for each observation are continuous quantities. When dealing with categorical independent variables, the equivalent technique is discriminant correspondence analysis.

II.iii. naive Bayes classifier
In machine learning, nave Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (nave) independence assumptions between the features. They are among the simplest Bayesian network models.
Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\overrightarrow{x} = (x_1, x_2, \ldots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $p(C_k|x_1, \ldots, x_n)$ for each of K possible outcomes or classes C_k.
The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as $p(C_k|\mathbf{x}) = \frac{\rho(c_k)p(\mathbf{x}|c_k)}{p(\mathbf{x})}$.

II.iv. binary tree
In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. A recursive definition using just set theory notions is that a (non-empty) binary tree is a tuple (L, S, R), where L and R are binary trees or the empty set and S is a singleton set.

III. Algorithm Implementation and Development

1. Use audacity to make a new piece of music with 5-second clip from 10 different songs sang by same band, convert the music to mono track, and save the music to wav file.
2. Load wav file in matlab by *audioread*, and convert the data of wav file to single- precision variable by *single*.
3. Create a spectrogram of wav file.
4. construct training data set.
5. construct test data set.



6. Use classify, fitctree, fitcnb to find classification.

IV. Computational Results

IV.i. test 1: Band Classification

I choose G-Dragon for K-POP, Hong Jin Young for Trot, and Wang Feng for Chinese Rock. The three bands are from different genres. The average accuracy for Naive Bayes is 64.17%, for LDA is 62.00%, for binary tree is 57.33%. The accuracy for Naive Bayes is the highest, and binary tree is the lowest. Naive Bayes and LDA are almost the same.

IV.i. test 2: The Case for Seattle

I choose G-dragon, Wanna One, and Turbo for k-pop. Because the time and location are almost the same, so the average accuracy should be almost the same. The average accuracy for Naive Bayes is 57.83%, for LDA is 53.33%, for binary tree is 49.50%. It can be seen from the data that the accuracy of Naive Bayes is the highest, while binary tree is the lowest.

IV.iii. test 3: Genre Classification

Test 3 is testing songs by different bands in three genres. The genres are k-pop, trot, and Chinese rock. The average accuracy for Naive Bayes is 44.00%, for LDA is 57.83%, for binary tree is 45.83%. It can be seen from the data that the accuracy of Naive Bayes and binary tree are almost the same, while LDA has higher accuracy.

V. Summary and Conclusions

In this assignment, I use LDA, Naive Bayes, and binary tree to identify music genre. Test 1 analysis 3 bands in different genres. Test 2 analysis 3 bands in the same genre - k-pop. Test 3 analysis 9 bands in 3 different genres. Except for test 3, Naive Bayes is the highest average accuracy, while binary tree is the lowest. In test 3, LDA is with the highest accuracy, while Naive Bayes and binary tree are almost the same.

Reference

1. Kutz, J. N. (n.d.). Data Driven Modeling & Scientific Computation
2. Makers of MATLAB and Simulink. (n.d.). Retrieved February 21, 2020, from *https* : *//www.mathworks.com/*

Appendix A. MATLAB functions used and brief implementation explanation

1. $[y, Fs] = audioread(filename)$: reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
2. $L = length(X)$: returns the length of the largest array dimension in X.
3. $r = unidrnd(n)$: generates random numbers from the discrete uniform distribution specified by its maximum value n.
4. $Y = floor(X)$: rounds each element of X to the nearest integer less than or equal to that element.
5. $X = ones(sz1, ..., szN)$ returns an sz1-by-...-by-szN array of ones where sz1,...,szN indicates the size of each dimension. For example, ones(2,3) returns a 2-by-3 array of ones.
6. $class = classify(sample, training, group)$ classifies each row of the data in sample into one of the groups in training. sample and training must be matrices with the same number of columns. group is a grouping variable for training.
7. $tree = fitctree(Tbl, Y)$ returns a fitted binary classification decision tree based on the input variables contained in the table Tbl and output in vector Y.
8. $label = predict(Mdl, X)$ returns a vector of predicted class labels for the predictor data in the table or matrix X, based on the trained, full or compact classification tree Mdl.
9. $Mdl = fitcnb(Tbl, Y)$ returns a multiclass naive Bayes model (Mdl), trained by the predictors in the table Tbl and class labels in the array Y.
10. $[s, f, t] = spectrogram(x, window, noverlap, f, fs)$ returns the spectrogram at the cyclical frequencies specified in f.
11. $[U, S, V] = svd(A)$ performs a singular value decomposition of matrix A, such that $A = U * S * V'$.

Appendix B. MATLAB codes

Code I: main file

```
clc;
clear all;
close all;

% load 9 wav files
[x1,Fs] = audioread('kpopgd.wav');
[x2,Fs] = audioread('kpopwannaone.wav');
[x3,Fs] = audioread('kpopturbo.wav');

[y1,Fs] = audioread('trothong.wav');
[y2,Fs] = audioread('trotpark.wav');
[y3,Fs] = audioread('trotlee.wav');

[z1,Fs] = audioread('rockwang.wav');
[z2,Fs] = audioread('rockcui.wav');
[z3,Fs] = audioread('rockluo.wav');

% add the wav file in same genre together
x = [x1;x2;x3];
y = [y1;y2;y3];
z = [z1;z2;z3];

L_x = length(x);
L_y = length(y);
L_z = length(z);

% Construct training dataset

num_training = 500;
num_test = 100;
number = 2;

accuracy_lda = [];
acuracy_nb = [];
accuracy_bt = [];

for kk = 1:5

    testnum = unidrnd(floor(0.8*L_x));

    % training data
    x_training = x([1:testnum-1, testnum+floor(0.2*L_x):end]);
```

```
    y_training = y([1:testnum-1, testnum+floor(0.2*L_y):end]);
    z_training = z([1:testnum-1, testnum+floor(0.2*L_z):end]);

    % test data
    x_test = x([testnum:testnum+floor(0.2*L_x)]);
    y_test = y([testnum:testnum+floor(0.2*L_y)]);
    z_test = z([testnum:testnum+floor(0.2*L_z)]);

    % get training data
    x_train = test_construct(num_training, x_training, length(x_training), Fs, number);
    y_train = test_construct(num_training, y_training, length(y_training), Fs, number);
    z_train = test_construct(num_training,z_training, length(z_training), Fs, number);

    % Plot clips

    clipind = 10000;

    figure(1)
%     subplot(3,1,1)
    plot(0:1/Fs:5,x(clipind:clipind+5*Fs))
    title('x')
%     subplot(3,1,2)
    figure(2)
    plot(0:1/Fs:5,y(clipind:clipind+5*Fs))
    title('y')
%     subplot(3,1,3)
    figure(3)
    plot(0:1/Fs:5,z(clipind:clipind+5*Fs))
    title('z')

    % Construct labels

    labels = [ones(num_training,1);2*ones(num_training,1);3*ones(num_training,1)];

    % Construct total training dataset

    training = abs([x_train';y_train';z_train']);

    % Construct test dataset

    test_labels = [ones(num_test,1);2*ones(num_test,1);3*ones(num_test,1)];

    x_test = test_construct(num_test, x_test, length(x_test), Fs, number);
    y_test = test_construct(num_test, y_test, length(y_test), Fs, number);
    z_test = test_construct(num_test, z_test, length(z_test), Fs, number);
```

```
    sample = abs([x_test';y_test';z_test']);

    % Classification

    class = classify(sample, training, labels);
    accuracy = sum(class==test_labels)/length(class);

    Mdl_ctree = fitctree(training, labels);
    class_ctree = predict(Mdl_ctree, sample);
    accuracy_ctree = sum(class_ctree==test_labels)/length(class);

    Mdl_cnb = fitcnb(training, labels);
    class_cnb = predict(Mdl_cnb, sample);
    accuracy_cnb = sum(class_cnb==test_labels)/length(class);

    accuracy_lda = [accuracy_lda; accuracy];
    accuracy_bt = [accuracy_bt; accuracy_ctree];
    acuracy_nb = [acuracy_nb; accuracy_cnb];
end

%

figure(4)
plot(accuracy_lda(1:5), '-');
hold on;
plot(accuracy_bt(1:5), '-');
plot(acuracy_nb(1:5), '-');
legend({'LDA','Binary Tree', 'Naive Bayes'},'Location','southeast');
xlabel('Number of Modes')
ylabel('Accuracy(%)')
title('Test 3')

figure(5)
% subplot(1,3,1)
pstart = 10000;
pend = pstart + 5*Fs;
clip = x(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
title('k-pop')

figure(6)
% subplot(1,3,2)
pstart = 10000;
pend = pstart + 5*Fs;
clip = y(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
```

```
title('trot')

figure(7)
% subplot(1,3,3)
pstart = 10000;
pend = pstart + 5*Fs;
clip = z(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
title('Chinese Rock')
```

Code II: test construct

```
function [trainingset] = test_construct(num, song, length, Fs, number)
    trainingset = [];
    for j = 1:num
        feature = [];
        pstart = unidrnd(length-5*Fs);
        pend = pstart + 5*Fs;
        clip = song(pstart:pend);
        [spec_clip] = spectrogram(clip,gausswin(500),200,[],Fs);
        [u,s,v] = svd(spec_clip,'econ');
        for j = 1:number
            feature = [feature;u(:,j)];
        end
        trainingset = [trainingset, feature];
    end
end
```