Hw4 : music classification

Author : Richy Yu

Abstract

This paper addresses classifying a given piece of music by sampling a 5 second clip. The tools used in this paper are machine learning, linear discriminant analysis. The programming software used is MATLAB.

Introduction and Overview:

The purpose of this paper is to write codes that can classify the music genre, by using the technique of machine learning algorithms. Technique of Linear Discriminant Analysis is used to create this identifier. There will be three tests. The first one is band classification. I will consider three different bands of different genres. The second one is the case for Seattle. In this case, choose three bands within the same genre. This makes the testing and separation much more challenging. Therefore, I expect to have lower accuracy. In the third case, I will use the same algorithms to broadly classify songs as jazz, rock, classic etc. In this case, I will choose three genres and build a classifier.

Theoretical Background

The singular value decomposition of a matrix A is defined as

$$A = U\Sigma V^*, \qquad (e.1)$$

Where U and V are unitary and Σ is diagonal. Diagonal values of Σ is ordered from largest to smallest. The following computations are extracted from our course notes to compute the singular value decomposition.

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

$$= V\Sigma^2 V^*$$

$$AA^T = (U\Sigma V^*)(U\Sigma V^*)^T$$

$$= U\Sigma V^* V\Sigma U^* \quad \text{(e.3)}$$

$$= U\Sigma^2 U^*$$

Multiplying (e.3) and (e.4) on the right by V and U respectively gives the two self-consistent eigenvalue problems

$$A^T AV = V\sum^2 \quad \text{(e.4)}$$

$$AA^T U = U\sum^2 \quad \text{(e.5)}$$

Thus if the normalized eigenvectors are found for these two equations, then the orthonormal basis vectors are produced for U and V. Likewise, the square root of the eigenvalues of these equations produces the singular values σ_j .

SVD is used for principle component analysis. Principle component analysis is to produce low-dimensional reductions of the dynamics and behavior when governing equations are not known.

Machine learning techniques will be used. The idea of machine learning is that given a training set, I will use an algorithm to identify the pattern in the training set and generalize the result to examples that are not seen in the training set. The techniques that are used is machine learning algorithms that can optimize the data and classify each music piece.

Linear Discriminant Analysis (LDA) is an algorithm that finds the linear combinations of the clear separation between the data distribution. One assumption that I learnt from my class of artificial intelligence is that the data must be linearly separable. In other words, there must exist a linear neural net that completely separate the data. Otherwise, the algorithm will not work. Other assumptions are the data is gaussian or normal. In addition, the variables are assumed to have constant variance. It 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. So LDA is a dimensionality reduction technique and it reduces the number of dimensions (i.e. variables) in a dataset while retaining

as much information as possible. 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 w such that

$$w = arg \max_{w} \frac{w^T S_B w}{w^T S_w w} \quad \text{(e.6)}$$

Where the scatter matrices for between-class S_B and within-class S_w data are given by

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \quad \text{(e.7)}$$

$$S_w = \sum_{j=1}^{2} \sum_x (x - \mu_j)(x - \mu_j)^T \quad \text{(e.8)}$$

These quantities essentially measure the variance of the data sets as well as the variance of the difference in the means. The criterion given by e.6 is commonly known as the generalized Rayleigh quotient whose solution can be found via the generalized eigenvalue problem

$$S_B w = \lambda S_w w \quad \text{(e.7)}$$

where the maximum eigenvalue λ and its associated eigenvector gives the quantity of interest and the projection basis.

All of these theoretical backgrounds are from our book. In MATLAB, I will not directly use these formulas.

Algorithm Implementation and Development

The first step is to import the music file using the command webread. The number of clips I choose is 40. However, these 40 clips are not from 40 different songs. Instead, I partition each song into several clips of 5 second length. For each clip, there exist both left and right channels, and I average the two datasets for each channel. Then I need to create spectrogram for the data file by using the Fourier transform and the method that I have used in previous homework. Then I reshape the spectrogram of frequencies. Repeat the same procedure for each band.

Before I write the trainer function, I aggregate the matrix for each band into one, for the purpose of creating a training set. I remember that I learn there is a condition for the training set. Indeed, each example has to appear constantly in the training set. By joining three matrix, I

guarantee that each example appear at least once. Then I write the trainer function. Just as the example in the class, the trainer function takes three bands and a feature variable. First, I perform the singular value decomposition on the bands. Then I project this onto principal components. Then I take 10 features for each class. I calculated the variance within the class and the variance between the classes of features. In this classification problem, there are three classes. Add up the distance between the centers of each cluster and the center of total data. We look for the maximum ratio of the between class scattering to the within class scattering, and multiply the coefficients with the three features. To get the thresholds, we trace each pair of computed values of two categories and get the average point of their bounds. The last step is testing, we pick new examples that are not in the training set. By comparing with the thresholds and the values of the examples, we make the classification and calculate the accuracy of the program.

Computational Results:

Again, the assumption of a successful training is that the data can be separated, so we assume the data are separable and perform the training.

Case one:

What we are interested in is the accuracy of the classification. For case one, in which genres and bands are all different, the accuracy I got is 0.776. I believe this is a pretty good result, because the classifier can correctly identify most of the song clips. This accuracy is believed to be the highest, because both bands and genres are different. It would be easy the classify.

Case two:

This case is believed to be more challenging to classify, because three bands are within the same genre. Since they are within the same genre, the songs will have many similarities and make it harder to classify. One trial of running the test gives an accuracy around 0.61. This

makes sense, because the songs are more difficult to separate. Suppose we have more training data, this result may be improved.

Case three:

In this case, we want to characterize the genre, and the bands may or may not be the same. One trial of running the test gives an accuracy around 0.7. The accuracy is between previous two cases. The songs of different genres are believed to have many difference. Our training algorithm may not have many difficulties to capture the features.

Summary and Conclusion :

As a summary, in this assignment I learn that machine learning can be used to identify the genre of a music. After running the algorithm on the training set, the algorithm will keep track of the characteristic of the examples in the set. If the data are linearly separable, we will be able to extend the result to other new examples that we have not seen in the training set. We are getting pretty good results overall.

Appendix A (MATLAB functions used) :

Size() : return the size of a matrix.
Fft: give the Fourier transform
Diag() : return a column vector of diagonal values.

Appendix B (MATLAB codes) :

```
clear; close all; clc

% Song/Music by artist Ketsa (Instrumental)
```

```
% urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_12_-
_Green_Man.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_11_-
_Slow_Vibing.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_07_-
_The_Stork.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_02_-
_Seeing_You_Again.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequency/Ketsa_-_08_-
_Multiverse.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_13_-
_Mission_Ready.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_10_-
_Memories_Renewed.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_09_-
_Life_Illusion.mp3"];

Ketsa_clips = [];  % 40 clips from the songs/music by Ketsa
dur = 5;  % 5 seconds clip
for i = 1:length(urls)
    [y, Fs] = webread(urls(i));
    for j = 1:5
        [y1, y2] = extract_clip2(y, Fs, 30 + j*20, dur);
        y_ave = (y1 + y2) / 2;
        Ketsa_clips = [Ketsa_clips y_ave];
    end
end
%%
[n,nclips] = size(Ketsa_clips);
L = n / Fs;  % 5 secs
t2 = linspace(0,L,n+1);
t = t2(1:n);
tslide = 0:0.1:L;
% All the spectrograms of the clips from Ketsa
Ketsa_dat = zeros(length(tslide) * n, nclips);
for i = 1:nclips  % for each clip, we create a corresponding spectrogram
    y = Ketsa_clips(:,i);
    v = y';
%     k = ((2*pi)/L)*[0:(n-1)/2 -(n-1)/2:-1];  % length of n is odd
%     ks = fftshift(k);
    vgt_spec = zeros(length(tslide), n);
    for j = 1:length(tslide)
        g = exp(-100*(t-tslide(j)).^2);
        vg = g.* v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    Ketsa_dat(:, i) = reshape(vgt_spec,length(tslide)*n,1);
end
```

```
%% Song/Music by artist Lately Kind of Yeah (Rock)

% urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_15_-_Heart_Feel.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_16_-_Johnny_Mathis.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_10_-_Tubescreamer.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_12_-_Kingdom_Come.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_02_-_Geist_IX.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_14_-_Goner.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_13_-_Two-Face.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_11_-_Long_Live.mp3"];

Lky_clips = [];  % 40 clips from the songs/music by Ketsa
dur = 5;  % 5 seconds clip
for i = 1:length(urls)
    [y, Fs] = webread(urls(i));
    for j = 1:5
        [y1, y2] = extract_clip2(y, Fs, 30 + j*20, dur);
        y_ave = (y1 + y2) / 2;
        Lky_clips = [Lky_clips y_ave];
    end
end

%%
[n,nclips] = size(Lky_clips);
L = n / Fs;  % 5 secs
t2 = linspace(0,L,n+1);
t = t2(1:n);
tslide = 0:0.1:L;
% All the spectrograms of the clips from Ketsa
Lky_dat = zeros(length(tslide) * n, nclips);
for i = 1:nclips  % for each clip, we create a corresponding spectrogram
    y = Lky_clips(:,i);
    v = y';
%     k = ((2*pi)/L)*[0:(n-1)/2 -(n-1)/2:-1];  % length of n is odd
%     ks = fftshift(k);
    vgt_spec = zeros(length(tslide), n);
    for j = 1:length(tslide)
        g = exp(-100*(t-tslide(j)).^2);
        vg = g.* v;
        vgt = fft(vg);
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
```

```
    Lky_dat(:, i) = reshape(vgt_spec,length(tslide)*n,1);
end

%% Song/Music by artist Dee Yan-Key (Latin)
% urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_06_-
_Beguine_Sailing_Trip.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_08_-
_Montuno_Evening_Mood.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_07_-
_Tango_Good_Air.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_10_-
_Fast_Bossa_Nova_Falling_Stars.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_11_-
_Rumba_Windblown.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_03_-
_Paso_Doble_Boat_Ahoy.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_05_-
_Habanera_By_the_Sea.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_01_-
_Salsa_After-Work_Party.mp3"];

Dyk_clips = [];  % 40 clips from the songs/music by Ketsa
dur = 5;  % 5 seconds clip
for i = 1:length(urls)
    [y, Fs] = webread(urls(i));
    for j = 1:5
        [y1, y2] = extract_clip2(y, Fs, 30 + j*30, dur);
        y_ave = (y1 + y2) / 2;
        Dyk_clips = [Dyk_clips y_ave];
    end
end

%%
[n,nclips] = size(Dyk_clips);
L = n / Fs;  % 5 secs
t2 = linspace(0,L,n+1);
t = t2(1:n);
tslide = 0:0.1:L;
% All the spectrograms of the clips from Ketsa
Dyk_dat = zeros(length(tslide) * n, nclips);
for i = 1:nclips  % for each clip, we create a corresponding spectrogram
    y = Dyk_clips(:,i);
    v = y';
%     k = ((2*pi)/L)*[0:(n-1)/2 -(n-1)/2:-1];  % length of n is odd
%     ks = fftshift(k);
    vgt_spec = zeros(length(tslide), n);
    for j = 1:length(tslide)
        g = exp(-100*(t-tslide(j)).^2);
        vg = g.* v;
        vgt = fft(vg);
```

```
        vgt_spec(j,:) = fftshift(abs(vgt));
    end
    Dyk_dat(:, i) = reshape(vgt_spec,length(tslide)*n,1);
end

%%
[U,S,V,w,vband1,vband2,vband3] = genre_trainer(Ketsa_dat,Lky_dat,Dyk_dat,10);

%%
plot(vband1,1,'ro');hold on
plot(vband2,2,'go');
plot(vband3,3,'bo');

sort1 = sort(vband1);
sort2 = sort(vband2);
sort3 = sort(vband3);

threshold1 = get_threshold(sort2,sort3);  % 7.0377e+03
threshold2 = get_threshold(sort3,sort1);  % 1.4806e+04

%% Test the classifier
% urls = ["https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_03_-
_Dusty_Hills.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_01_-
_Dreaming_Days.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Creative_Commons/Ketsa/Raising_Frequecy/Ketsa_-_05_-
_Crescents.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_14_-_Goner.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_01_-_Specific_Ocean.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_03_-_The_Little_Red_School_House.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_12_-
_Samba_Pop_Pancake_Tuesday.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_04_-
_Samba_Rose_Monday.mp3";
%     "https://files.freemusicarchive.org/storage-freemusicarchive-
org/music/ccCommunity/Dee_Yan-Key/latin_summer/Dee_Yan-Key_-_09_-_Cha-Cha-
Cha_Beach_Life.mp3"];

test_clips = [];  % 40 clips from the songs/music by Ketsa
dur = 5;  % 5 seconds clip
slice_time = [30 30 30 20 20 10 35 40 40];
for i = 1:length(urls)
    [y, Fs] = webread(urls(i));
    for j = 1:5
        [y1, y2] = extract_clip2(y, Fs, 30 + j*slice_time(i), dur);
        y_ave = (y1 + y2) / 2;
```

```
        test_clips = [test_clips y_ave];
    end
end

%%
TestNum = size(test_clips,2)
Test_wave = wavelet_spectrogram(test_clips, Fs); % wavelet transformation
TestMat = U'*Test_wave;  % PCA projection
pval = w'*TestMat;  % LDA projection
%%
hidden_labels = repelem([1 2 3], 15); % Answer to our test data

ResVec = zeros(1,TestNum); % Answer by the classifier model
for i = 1:TestNum
    if pval(i) > threshold2
        ResVec(i) = 1; % band1
    elseif pval(i) < threshold1
        ResVec(i) = 2; % band2
    else
        ResVec(i) = 3; % band3
    end
end

err_num = 0;
for i = 1:TestNum
    if (ResVec(i) ~= hidden_labels(i))
        err_num = err_num + 1;
    end
end

disp('Number of mistakes')
err_num
disp('Rate of success');
sucRate = 1-err_num/TestNum

%%
function musData = wavelet_spectrogram(clips_data, Fs)
    [n,nclips] = size(clips_data);
    L = n / Fs;  % 5 secs
    t2 = linspace(0,L,n+1);
    t = t2(1:n);
    tslide = 0:0.1:L;
    % All the spectrograms of the clips from Ketsa
    musData = zeros(length(tslide) * n, nclips);
    for i = 1:nclips  % for each clip, we create a corresponding spectrogram
        y = clips_data(:,i);
        v = y';
    %     k = ((2*pi)/L)*[0:(n-1)/2 -(n-1)/2:-1];  % length of n is odd
    %     ks = fftshift(k);
        vgt_spec = zeros(length(tslide), n);
        for j = 1:length(tslide)
            g = exp(-100*(t-tslide(j)).^2);
            vg = g.* v;
            vgt = fft(vg);
            vgt_spec(j,:) = fftshift(abs(vgt));
        end
        musData(:, i) = reshape(vgt_spec,length(tslide)*n,1);
```

```
    end
end


% Load a clip of a song with a specifed url, start time and duration in
% seconds. Assume the input time in the range of the duration of the song.
function [y1, y2, Fs] = extract_clip(url, start_time, duration)

    % y is an N x 2 array: the first column represents the left channel
    % and the second column represents the right channel.
    [y, Fs] = webread(url);
    mus_length = length(y) / Fs;  % duration of the song in seconds
    if (start_time > mus_length) || (start_time + duration > mus_length)
        disp(url)
        error("ERROR.\nThe specified range of the clip exceeds the length of
the song")
    end
    start_point = start_time * Fs;
    end_point = start_point + duration * Fs;
    y = y(start_point:end_point, :);
    y1 = y(:,1); y2 = y(:,2);

end


% 2nd ver: Load a clip based on input signal.
function [y1, y2] = extract_clip2(y, Fs, start_time, duration)

    % y is an N x 2 array: the first column represents the left channel
    % and the second column represents the right channel.
    mus_length = length(y) / Fs;  % duration of the song in seconds
    if (start_time > mus_length) || (start_time + duration > mus_length)
        error("ERROR: The specified range of the clip exceeds the length of
the song")
    end
    start_point = start_time * Fs;
    end_point = start_point + duration * Fs;
    y = y(start_point:end_point, :);
    y1 = y(:,1); y2 = y(:,2);

end


function [U,S,V,w,vband1,vband2,vband3] =
genre_trainer(band1,band2,band3,feature)
    n1 = size(band1,2);
    n2 = size(band2,2);
    n3 = size(band3,2);
    [U,S,V] = svd([band1 band2 band3],'econ');
    genres = S*V'; % projection onto principal components
    U = U(:,1:feature);
    band11 = genres(1:feature,1:n1);
    band22 = genres(1:feature,n1+1:n1+n2);
    band33 = genres(1:feature,n1+n2+1:n1+n2+n3);

    mband1 = mean(band11,2);
```

```
    mband2 = mean(band22,2);
    mband3 = mean(band33,2);
    mtotal = mean([mband1 mband2 mband3],2);

    Sw = 0; % within class variances
    for k = 1:n1
        Sw = Sw + (band11(:,k)-mband1)*(band11(:,k)-mband1)';
    end
    for k = 1:n2
        Sw = Sw + (band22(:,k)-mband2)*(band22(:,k)-mband2)';
    end
    for k = 1:n3
        Sw = Sw + (band33(:,k)-mband3)*(band33(:,k)-mband3)';
    end
    Sb = 40*(mband1-mtotal)*(mband1-mtotal)' + 40*(mband2-mtotal)*(mband2-
mtotal)'...
        + 40*(mband3-mtotal)*(mband3-mtotal)';

    [V2,D] = eig(Sb,Sw); % linear discriminant analysis
    [~,ind] = max(abs(diag(D)));
    w = V2(:,ind); w = w/norm(w,2);

    vband1 = w'*band11;
    vband2 = w'*band22;
    vband3 = w'*band33;

end


% method helps to determine a threshold between two adjacent categories
% Assume values of s1 are just lower than those of s2
function threshold = get_threshold(s1, s2)
    t1 = length(s2);
    t2 = 1;
    while s2(t1) > s1(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold = (s2(t1)+s1(t2))/2;
end
```