Music Genre Classification

Yuanheng (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 intentially separates, two or more classes of objects. It focuses on maximizing the seperatibility 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 ***w*** such that

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

where

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

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

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

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \tag{2.1.4}$$

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 S_B becomes the summation of distances between the center of the total scatters and the center of each category.

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

where m is the number of classes, μ 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: threshold1 = 7.0377e+03 and threshold2 = 1.4806e+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 band2 < band3 < 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.

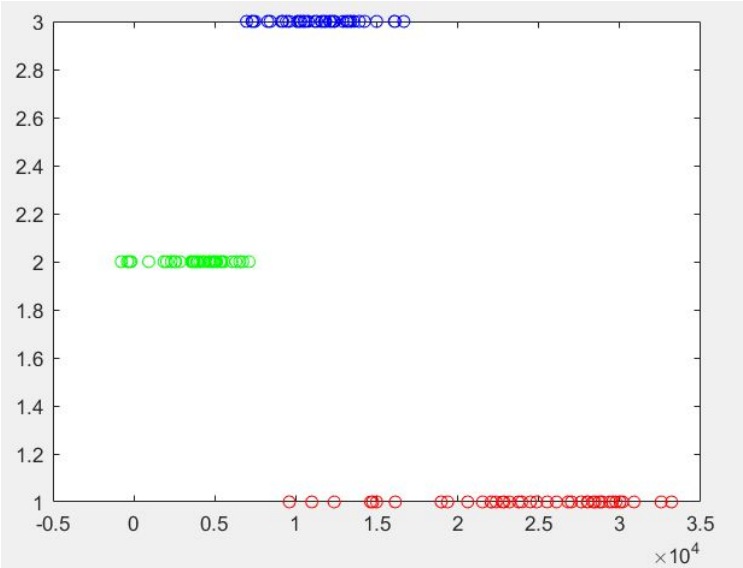

Figure 1: LDA projection of three classes

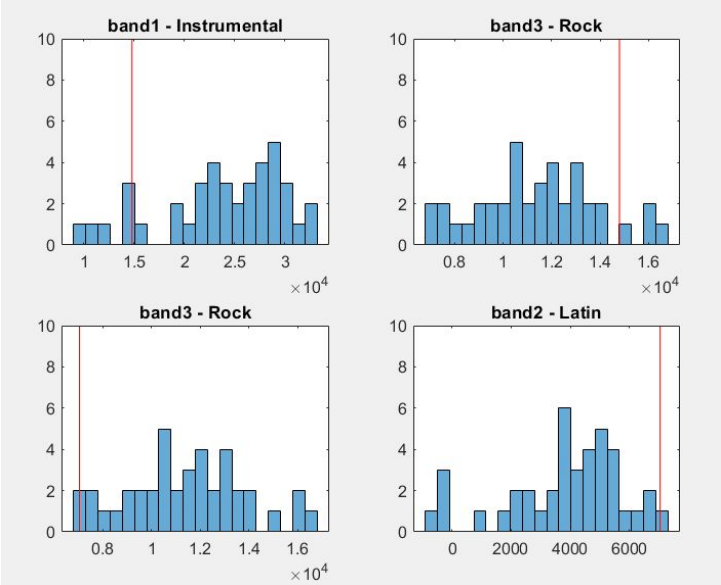

Figure 2: Histogram of the statistics associated with the three classes projected onto the LDA basis

Appendix A: MATLAB functions used

1) webread: Read content from RESTful web service
2) svd: produces an economy-size decomposition of m-by-n matrix A.
3) repelem: Repeat copies of array elements.

Appendix B: MATLAB codes

```
%% Song/Music by artist Ketsa (Instrumental)
urls =
["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_12_-_Green_Man.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_11_-_Slow_Vibing.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_07_-_The_Stork.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_02_-_Seeing_You_Again.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequency/Ketsa_-_08_-_Multiverse.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_13_-_Mission_Ready.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_10_-_Memories_Renewed.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/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
%%
Ketsa_dat = wavelet_spectrogram(Ketsa_clips, Fs);

%% 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

%%
Lky_dat = wavelet_spectrogram(Lky_clips, Fs);

%% 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

%%
Dyk_dat = wavelet_spectrogram(Dyk_clips, Fs);

%%
[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
%%
figure
subplot(2,2,1)
histogram(sort1,20); hold on, plot([threshold2 threshold2], [0 10], 'r')
title('band1 - Instrumental')
subplot(2,2,2)
histogram(sort3,20); hold on, plot([threshold2 threshold2], [0 10], 'r')
title('band3 - Rock')
subplot(2,2,3)
histogram(sort3,20); hold on, plot([threshold1 threshold1], [0 10], 'r')
title('band3 - Rock')
subplot(2,2,4)
histogram(sort2,20); hold on, plot([threshold1 threshold1], [0 10], 'r')
title('band2 - Latin')


%% Test the classifier
urls =
["https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_03_-_Dusty_Hills.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_01_-_Dreaming_Days.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Creative_Commons/Kets
a/Raising_Frequecy/Ketsa_-_05_-_Crescents.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/Latel
y_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah_-_14_-_Goner.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/Latel
y_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah_-_01_-_Specific_Ocean.mp3";

"https://files.freemusicarchive.org/storage-freemusicarchive-org/music/Decoder_Magazine/Latel
y_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
% Report the testing result
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;
    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
```