Ruimin Zhang
March 6th, 2020

Music Classification Using LDA

Abstract:

This is the homework 4 of AMATH 482 in the University of Washington. This assignment focuses on classifying different music clips.

Sec. I. Introduction and Overview

This assignment contains three parts with similar algorithm. Test 1 is band classification where we are asked to consider three artists of different genre. Test 2 is the case where three artists are of the same genre. Test 3 is when we consider multiple artists from different genres. Even though they may sound differently, but the algorithm is very similar.

I will first discuss the theoretical background used in this assignment. Then I will show the algorithm and my classification results.

Sec. II. Theoretical Background

2.1 LDA

The idea of the Linear Discrimination Analysis(LDA) was first proposed in the context of taxonomy. In our example, three data sets are considered and projected onto new bases. In the left figure, the projection shows the data to be completely mixed, not allowing for any reasonable way to separate the data from each other. In the right figure, which is the ideal caricature for LDA, the data are well separated with the means μ1, μ2 and μ3 being well apart when projected onto the chosen subspace. Suppose μ is the mean of all three data sets.

Thus 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

$$\mathbf{w} = \text{argmax}_{\mathbf{w}} \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}} \tag{1}$$

where the scatter matrices for between-class $\mathbf{S}_{\mathrm{B}}$ and within-class $\mathbf{S}_{\mathrm{W}}$ data are given by

$$\mathbf{S}_{\mathrm{B}} = \sum_{j=1}^{3}(\mu_j - \mu)(\mu_j - \mu)^T \tag{2}$$

$$\mathbf{S}_{\mathrm{W}} = \sum_{\mathrm{j}=1}^{2}\sum_{\mathbf{x}}(\mathbf{x} - \mu_j)(\mathbf{x} - \mu_j)^{\mathrm{T}}. \tag{3}$$

These quantities essentially measure the variance of the data sets as well as the variance of the difference in the means. There are a number of tutorials available online with details of the method, thus the mathematical details will not be presented here. The criterion given by Eq. (1) is commonly known as the generalized Rayleigh quotient whose solution can be found via the generalized eigenvalue problem

$$\mathbf{S}_{\mathrm{B}}\mathbf{w} = \lambda\mathbf{S}_{\mathrm{W}}\mathbf{w} \tag{4}$$

where the maximum eigenvalue λ and its associated eigenvector gives the quantity of interest and the projection basis. Thus once the scatter matrices are constructed, the generalized eigenvectors can easily be constructed with MATLAB.

Sec. III. Algorithm Implementation and Development

The mathematical objective is to develop an algorithm by which we can train the computer to distinguish between dogs and cats. The algorithm will consist of the following key steps:

• Step 1: Transform music clips into spectrograms. The motivation is simply to provide an effective method for doing edge detection. The training sets for an artist/genre will be (the number of songs) $\times$ (the number of clips in each song).

• Step 2: From the spectrogram images, find the principal components associated with each artist or genre respectively. Project the images onto the SVD/PCA/POD modes selected from the training algorithm. I decide to preserve the first 26 columns(features).

• Step 3: Design a statistical decision threshold for discrimination between different types of artists/genre. The method to be used here will be a linear discrimination analysis (LDA). Project this two-level decomposition onto the LDA eigenvector. Determine the projection value relative to the LDA threshold determined in the training algorithm.

As for determining thresholds, if we see pattern such that (majority of data 2) $\leq$ (majority of data 3) $\leq$ (majority of data 1), then the two thresholds are between 2 & 3 and between 3 & 1. We define the threshold to be the location where the number of wrong data are equal on both sides of the threshold.

• Step 4: Classify three songs. Test the algorithm efficiency and accuracy by running through several clips of three songs. Then we compare with the real classifications of the clips and determine the accuracy of our classification.

Sec. IV. Computational Results

For test1, I included three artists (or bands) sodagreen, (g)-idle, and KUN from the genres of Taiwanese indie music, hip-hop, and R&B, respectively. I used six songs from each artists (or bands) as training data. For test2, I specifically targeted at the genre of K-pop and used a total of 18 songs of three girl groups – blackpink, (g)-idle, and mamamoo. For test3, I used three genres – K-pop, classical, and R&B – and included five songs for each genre which are derived from at least three artists.

Test 1: Band Classification of Different Genres

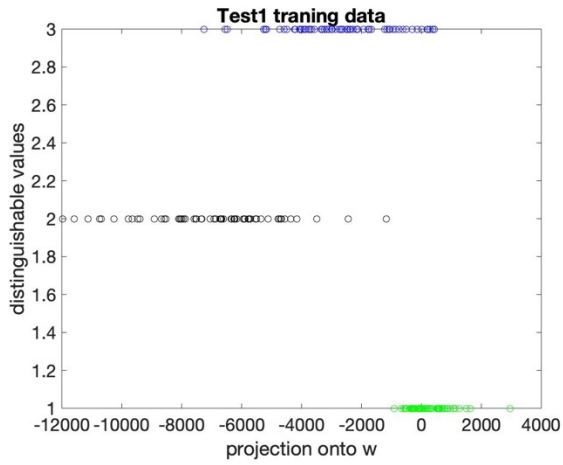

Figure1. data projected on w is differentiated into different genres. The green dots of value 1 is the training data from the first artist (or band); the black dots of value 2 is the training data from the second artist (or band); the blue dots of value 2 is the training data from the third artist (or band).

Figure2. the magenta stars are the projected values of the first artist (or band); the cyan stars are the projected values of the second artist (or band); the red stars are the projected values of the third artist (or band). The dotted line is the first threshold to distinguish artist 2 and artist 3, whereas the solid line is the second threshold to distinguish artist 3 and artist 1.

Testing results:
Number of mistakes: errNum = 11
Rate of success: sucRate = 0.6667

This is reasonable because the genre I selected are pretty close to each other. It is natural if some of the features looks similar to each other. If I am asked to revise, I would try some really different genres.

Test 2: Band Classification of Same Genre

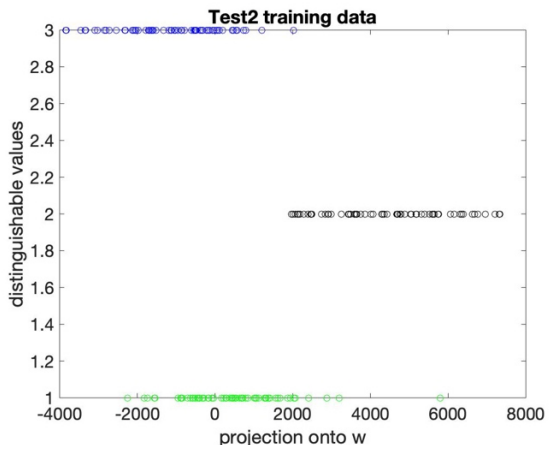

Figure3. data projected on w is differentiated into different genres. The green dots of value 1 is the training data from the first artist (or band); the black dots of value 2 is the training data from the second artist (or band); the blue dots of value 2 is the training data from the third artist (or band).

Figure4. the magenta stars are the projected values of the first artist (or band); the cyan stars are the projected values of the second artist (or band); the red stars are the projected values of the third artist (or band). The dotted line is the first threshold to distinguish artist 2 and artist 3, whereas the solid line is the second threshold to distinguish artist 3 and artist 1.

Testing results:
Number of mistakes: errNum = 15
Rate of success: sucRate = 0.5455

The rate of success is low because within the same genre, it is often difficult to distinguish the features. By observation, we notice that the testing data of the second artist is out of the bound. I assume this is because my training data is not diverse enough.

Test 3: Genre Classification

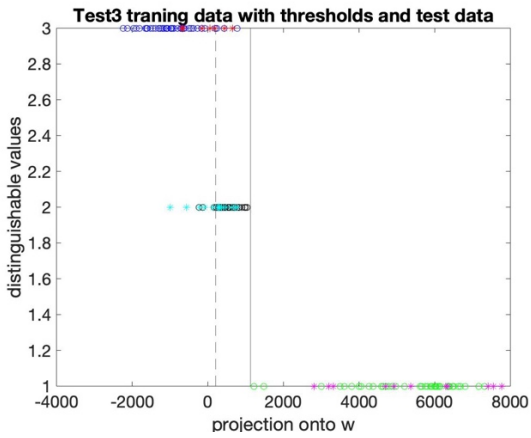

Figure5. data projected on w is differentiated into different genres. The green dots of value 1 is the training data from the first genre the black dots of value 2 is the training data from the second genre; the blue dots of value 2 is the training data from the third genre.

Figure6. the magenta stars are the projected values of the first genre the cyan stars are the projected values of the second genre the red stars are the projected values of the third genre The dotted line is the first threshold to distinguish genre 2 and genre 3, whereas the solid line is the second threshold to distinguish genre 3 and genre 1.

Testing results:
Number of mistakes: errNum = 5
Rate of success: sucRate = 0.8485

This result is good because we assume the rate of success is high since the data should be pretty distinguishable.

Sec. V. Summary and Conclusions
To sum up, depending on which test we are running, we first obtain some songs of different artists or genre. Then we select several clips from them as the training data. Then we transform the data to spectrograms so that we can perform SVD. After deriving the principle component of those clips, we use LDA to distinguish the features between clips of different artists or genre. Finally, we use some new clips to test how accurate our classification results are.

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. `s = spectrogram(x)` returns the short-time Fourier transform of the input signal, `x`. Each column of `s` contains an estimate of the short-term, time-localized frequency content of `x`.

3. `B = reshape(A,sz)` reshapes `A` using the size vector, `sz`, to define `size(B)`. For example, `reshape(A,[2,3])` reshapes `A` into a 2-by-3 matrix. `sz` must contain at least 2 elements, and `prod(sz)` must be the same as `numel(A)`.

4. `[V,D] = eig(A,B)` returns diagonal matrix `D` of generalized eigenvalues and full matrix `V` whose columns are the corresponding right eigenvectors, so that `A*V = B*V*D`.

5. `B = sort(A)` sorts the elements of `A` in ascending order.
   - If `A` is a vector, then `sort(A)` sorts the vector elements.
   - If `A` is a matrix, then `sort(A)` treats the columns of `A` as vectors and sorts each column.
   - If `A` is a multidimensional array, then `sort(A)` operates along the first array dimension whose size does not equal 1, treating the elements as vectors.
6. `xline(xvalue)` creates a constant vertical line at the specified *x*-value in the current axes. For example, `xline(2)` creates a line at `x = 2`.

Appendix B.MATLAB codes

```
%test 1
```

```
clear; close all; clc

% artist 1: sodagreen(first 6 songs)
% artist 2: (g)-idle (middle 6 songs)
% artist 3: KUN (last 6 songs)
filename =
['miss.mp3';'rain.mp3';'meet.mp3';'walk.mp3';'lone.mp3';'song.mp3';

'blow.mp3';'lata.mp3';'seno.mp3';'hann.mp3';'lion.mp3';'uhoh.mp3';

'yong.mp3';'rebi.mp3';'pull.mp3';'iwgl.mp3';'snow.mp3';'wait.mp3'];

test = [];
sec = [10 20 30 40 50 60 70 80 90 100 110];
for k = 1:size(filename,1)
   [y,Fs] = audioread(filename(k,:));
    y = y(:,1);
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i))); p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        test = [test; Sp];
    end
end

%%
test = test';
feature=26; % 1<feature<length(sec)*size(filename,1)
[U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature);

% plot our projected data and threshold
plot(sort1,1,'go'),hold on
plot(sort2,2,'ko')
plot(sort3,3,'bo')
xlabel('projection onto w'), ylabel('distinguishable values')
title('Test1 traning data')
set(gca,'Fontsize',16)
%% create threshold based on previous graph
[t1,t2] = threshold(sort2,sort3,sort1);
xline(t1,'--');
xline(t2,'-');

%% Test classifier
filename2 = ['SMAL.mp3';'NAME.mp3';'HTG_.mp3'];
TestSet = [];
sec = [10 20 30 40 50 60 70 80 90 100 110];
for k = 1:size(filename2,1)
    [y,Fs] = audioread(filename2(k,:));
    y = y(:,1);
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i))); p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        TestSet = [TestSet; Sp];
    end
end
```

```
TestSet = TestSet';
%%
label1 = ones(1,length(sec));
label2 = ones(1,length(sec)) * 2;
label3 = ones(1,length(sec)) * 3;
hiddenlabels = [label1, label2, label3];

TestMat = U'*TestSet;  % PCA projection
pval = w'*TestMat;  % LDA projection

%% Let's find out accuracy!

r = [];
for i = 1:(size(filename2,1)*length(sec))
    if pval(i)<=t1
        r = [r; 2];
    elseif pval(i) <=t2
        r = [r; 3];
    else
        r = [r; 1];
    end
end
r = r';

disp('Number of mistakes')
errNum = sum(abs(r-hiddenlabels)~=0)

disp('Rate of success');
sucRate = 1-errNum/(size(filename2,1)*length(sec))
%%
% [~,~,~,~,sortA,sortB,sortC] = dc_trainer(TestSet,26);

    plot(pval(1:11),1,'m*'),hold on
    plot(pval(12:26),2,'c*')
    plot(pval(27:end),3,'r*')

    title('Test1 traning data with thresholds and test data')

%%
function [threshold1,threshold2] = threshold(sort1,sort2,sort3)
    t1 = length(sort1);
    t2 = 1;
    while sort1(t1)>sort2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sort1(t1)+sort2(t2))/2;

    t1 = length(sort2);
    t2 = 1;
    while sort2(t1)>sort3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sort2(t1)+sort3(t2))/2;
end
```

```
function [U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature)
    n1 = size(test,2)/3; n2 = size(test,2)/3;n3 = size(test,2)/3;

    [U,S,V] = svd(test,'econ');

    bands = S*V'; % projection onto principal components
    U = U(:,1:feature);

    band1 = bands(1:feature,1:n1);
    band2 = bands(1:feature,n1+1:n1+n2);
    band3 = bands(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(band1,2);
    m2 = mean(band2,2);
    m3 = mean(band3,2);
    totalm = mean(bands,2);
    totalm = totalm(1:feature,:);% why?

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (band1(:,k)-m1)*(band1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (band2(:,k)-m2)*(band2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (band3(:,k)-m3)*(band3(:,k)-m3)';
    end

     % between class
    Sb = (m1-totalm)*(m1-totalm)'+(m2-totalm)*(m2-totalm)'+(m3-totalm)*(m3-
totalm)';

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

    v1 = w'*band1;
    v2 = w'*band2;
    v3 = w'*band3;

    sort1 = sort(v1);
    sort2 = sort(v2);
    sort3 = sort(v3);

%     plot(sort1,1,'go'),hold on
%     plot(sort2,2,'ko')
%     plot(sort3,3,'bo')

end
```

```
% test2
clear; close all; clc

% artist 1: blackpink (first 6 songs)
% artist 2: (g)-idle (middle 6 songs)
% artist 3: mamamoo (last 6 songs)
filename =
['1bmby.mp3';'1dddd.mp3';'1fire.mp3';'1kill.mp3';'1last.mp3';'1whis.mp3';

'2wind.mp3';'2star.mp3';'2hip_.mp3';'2glea.mp3';'2gbb_.mp3';'2deca.mp3'; %
2ego_

'3blow.mp3';'3pits.mp3';'3seno.mp3';'3hann.mp3';'3lion.mp3';'3uhoh.mp3']; %
test = [];
sec = [10 20 30 40 50 60 70 80 90 100 110];
for k = 1:size(filename,1)
   [y,Fs] = audioread(filename(k,:));
    y = y(:,1);
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i))); p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        test = [test; Sp];
    end
end

%%
test = test';
feature=26; % 1<feature<length(sec)*size(filename,1)
[U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature);

% plot our projected data and threshold
plot(sort1,1,'go'),hold on
plot(sort2,2,'ko')
plot(sort3,3,'bo')

xlabel('projection onto w'), ylabel('distinguishable values')
title('Test2 training data')
set(gca,'Fontsize',16)

%% create threshold based on previous graph
[t1,t2] = threshold(sort3,sort1,sort2);
xline(t1,'--');
xline(t2,'-');


%% Test classifier
filename2 = ['1whis.mp3';'2ego_.mp3';'3lata.mp3'];
TestSet = [];
sec = [30 40 50 60 70 80 90 100 110 120 130];
for k = 1:size(filename2,1)
    [y,Fs] = audioread(filename2(k,:));
    y = y(:,1);
```

```
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i))); p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        TestSet = [TestSet; Sp];
    end
end

TestSet = TestSet';
%%
label1 = ones(1,length(sec));
label2 = ones(1,length(sec)) * 2;
label3 = ones(1,length(sec)) * 3;
hiddenlabels = [label1, label2, label3];

TestMat = U'*TestSet;  % PCA projection
pval = w'*TestMat;  % LDA projection

%% Let's find out accuracy!

r = [];
for i = 1:(size(filename2,1)*length(sec))
    if pval(i)<=t1
        r = [r; 3];
    elseif pval(i) <=t2
        r = [r; 1];
    else
        r = [r; 2];
    end
end
r = r';

disp('Number of mistakes')
errNum = sum(abs(r-hiddenlabels)~=0)

disp('Rate of success');
sucRate = 1-errNum/(size(filename2,1)*length(sec))
%%

    plot(pval(1:11),1,'m*'),hold on
    plot(pval(12:26),2,'c*')
    plot(pval(27:end),3,'r*')

    title('Test1 traning data with thresholds and test data')

%%
function [threshold1,threshold2] = threshold(sort1,sort2,sort3)
    t1 = length(sort1);
    t2 = 1;
    while sort1(t1)>sort2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sort1(t1)+sort2(t2))/2;

    t1 = length(sort2);
```

```
    t2 = 1;
    while sort2(t1)>sort3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sort2(t1)+sort3(t2))/2;
end


function [U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature)
    n1 = size(test,2)/3; n2 = size(test,2)/3;n3 = size(test,2)/3;

    [U,S,V] = svd(test,'econ');

    bands = S*V'; % projection onto principal components
    U = U(:,1:feature);

    band1 = bands(1:feature,1:n1);
    band2 = bands(1:feature,n1+1:n1+n2);
    band3 = bands(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(band1,2);
    m2 = mean(band2,2);
    m3 = mean(band3,2);
    totalm = mean(bands,2);
    totalm = totalm(1:feature,:);% why?

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (band1(:,k)-m1)*(band1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (band2(:,k)-m2)*(band2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (band3(:,k)-m3)*(band3(:,k)-m3)';
    end

     % between class
    Sb = (m1-totalm)*(m1-totalm)'+(m2-totalm)*(m2-totalm)'+(m3-totalm)*(m3-
totalm)';

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

    v1 = w'*band1;
    v2 = w'*band2;
    v3 = w'*band3;

    sort1 = sort(v1);
    sort2 = sort(v2);
    sort3 = sort(v3);

%     plot(sort1,1,'go'),hold on
%     plot(sort2,2,'ko')
```

```
%     plot(sort3,3,'bo')

end
```

```
%test3
clear; close all; clc
% genre 1: k-pop (blackpink, gidle, mamamoo)
% % % genre 2: classical (Dee Yan-Key, Lobo Loco, Chad Crouch, Yakov Golman)
% genre 3: RnB (Vincent Augustus, Glad Rags, Checkie Brown, Florian Decros)

filename = ['1bmby.mp3';'1star.mp3';'1hip_.mp3';'1kill.mp3';'1lata.mp3';
            '2todo.mp3';'2swan.mp3';'2jaun.mp3';'2newy.mp3';'2PPSA.mp3';
            '3kuge.mp3';'3soka.mp3';'3viol.mp3';'3shes.mp3';'3sunt.mp3'];

test = [];
sec = [10 20 30 90 100 110 120 130];
for k = 1:size(filename,1)
   [y,Fs] = audioread(filename(k,:));
    y = y(:,1);
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i)));p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        test = [test; Sp];
    end
end
test = test';

%%
feature=26; % 1<feature<length(sec)*size(filename,1)
[U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature);

% plot our projected data and threshold
plot(sort1,1,'go'),hold on
plot(sort2,2,'ko')
plot(sort3,3,'bo')

%% create threshold based on previous graph
[t1,t2] = threshold(sort3,sort2,sort1);
xline(t1,'--');
xline(t2,'-.');
xlabel('projection onto w'), ylabel('distinguishable values')
title('Test 3')
set(gca,'Fontsize',16)
```

```
%% Test classifier
filename2 = ['1ego_.mp3';'2fore.mp3';'3UTU_.mp3'];
TestSet = [];
sec = [30 40 50 60 70 80 90 100 110 120 130];
for k = 1:size(filename2,1)
    [y,Fs] = audioread(filename2(k,:));
    y = y(:,1);
    for i = 1:length(sec)
        p = y(sec(i)*Fs:Fs*(5+sec(i))); p=p';
        Sp =abs(spectrogram(p));
        Sp = reshape(Sp, [1,8*32769]);
        TestSet = [TestSet; Sp];
    end
end

TestSet = TestSet';
%%
label1 = ones(1,length(sec));
label2 = ones(1,length(sec)) * 2;
label3 = ones(1,length(sec)) * 3;
hiddenlabels = [label1, label2, label3];

TestMat = U'*TestSet;  % PCA projection
pval = w'*TestMat;  % LDA projection

%% Let's find out accuracy!

r = [];
for i = 1:(size(filename2,1)*length(sec))
    if pval(i)<=t1
        r = [r; 3];
    elseif pval(i) <=t2
        r = [r; 2];
    else
        r = [r; 1];
    end
end
r = r';

disp('Number of mistakes')
errNum = sum(abs(r-hiddenlabels)~=0)

disp('Rate of success');
sucRate = 1-errNum/(size(filename2,1)*length(sec))
%%
[~,~,~,~,sortA,sortB,sortC] = dc_trainer(TestSet,26);

    plot(sortA,1,'m*'),hold on
    plot(sortB,2,'c*')
    plot(sortC,3,'r*')

    legend({'song1','song2','song3','threshold1','threshold2'})

%%
function [threshold1,threshold2] = threshold(sort1,sort2,sort3)
```

```
    t1 = length(sort1);
    t2 = 1;
    while sort1(t1)>sort2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (sort1(t1)+sort2(t2))/2;

    t1 = length(sort2);
    t2 = 1;
    while sort2(t1)>sort3(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold2 = (sort2(t1)+sort3(t2))/2;
end


function [U,S,V,w,sort1,sort2,sort3] = dc_trainer(test,feature)
    n1 = size(test,2)/3; n2 = size(test,2)/3;n3 = size(test,2)/3;

    [U,S,V] = svd(test,'econ');

    bands = S*V'; % projection onto principal components
    U = U(:,1:feature);

    band1 = bands(1:feature,1:n1);
    band2 = bands(1:feature,n1+1:n1+n2);
    band3 = bands(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(band1,2);
    m2 = mean(band2,2);
    m3 = mean(band3,2);
    totalm = mean(bands,2);
    totalm = totalm(1:feature,:);% why?

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (band1(:,k)-m1)*(band1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (band2(:,k)-m2)*(band2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (band3(:,k)-m3)*(band3(:,k)-m3)';
    end

     % between class
    Sb = (m1-totalm)*(m1-totalm)'+(m2-totalm)*(m2-totalm)'+(m3-totalm)*(m3-
totalm)';

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

    v1 = w'*band1;
```

```
    v2 = w'*band2;
    v3 = w'*band3;

    sort1 = sort(v1);
    sort2 = sort(v2);
    sort3 = sort(v3);

%     plot(sort1,1,'go'),hold on
%     plot(sort2,2,'ko')
%     plot(sort3,3,'bo')

end
```

Reference

Kutz, Nathan. *Data Driven Modeling & Scientific Computing*