# AMATH 482: Implementing a Music Classifier Program on MATLAB

Lahari Gorantla

## Abstract

Most humans can recognize the different genres in music instantly by ear, whether it be the sweet blues, soft classical music, or charming rock. The motivation behind this paper is to test the accuracy of a computer program in classifying different music into their respective genres by training it with multiple 5-second data sets and then testing with a test set. To put the program to the test, there will be 3 test cases: band classification, KPOP-based band classifications given a genre, and general genre classification.

## I. Introduction and Overview

Different genres and artists have their own unique signature style. Even after hearing a brief instance of a song, most humans can categorize the music into a genre, artist, or some other group-able characteristic. The question is whether a computer program can perform this task with high accuracy given a 5 second sample of a musical clip. This concept of machine learning tests the computer's prediction accuracy based of a *testing set* after being built with a *training set.* The goal of the supervised learning algorithm here is to utilize dimensionality reduction technique, principal component analysis (PCA), with singular value decomposition (SVD) to represent the data. The ability for a computer to make distinctions based of previous knowledge is revolutionary in our modern world and has roots in many innovative applications.

For this lab report, 3 main test cases were given. The first test case was band classification, where 3 different bands from 3 different genres were chosen. The second test case was choosing 3 bands within the same genre. The third case was building an algorithm that can identify between 3 different genres. In all cases 90 5-second samples for each test case were to train the program (it breaks down to 30 samples for each of the 3 bands/artists/genres) and then a testing set of 30 5-second samples were used to test the accuracy (it breaks down to 10 sample for each category). To understand the accuracy, linear discriminant analysis (LDA), statistical analytic methods, and PCA projection methods are used to describe the datasets.

## II. Theoretical Background

**Spectrogram with Gabor Filter**

A spectrogram is the visual representation of the time and frequency domain data, where the color represents the intensity of the signal strength. The Gabor transform, also known as short-time Fourier Transform, is a modification of the Fourier transform equation where the term $g(\tau-t)$ acts as a time filter to localize the signal over a specific window of time. The integration of this term slides the window down the signal to localize time chunks to decompose it into its respective frequencies. The term $\omega$ denotes the angular frequency of the signal, and the term $\tau$ slides the window.

$$G[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau \qquad (1)$$

Because musical pieces are dynamic over a time course, taking the Fourier transform of the whole piece would merely average it and lose a lot of important information about its dynamics. A spectrogram with a Gabor transform performs a frequency-time analysis of each piece.

**Principal Component Analysis (PCA)**

PCA is a linearity dimensionality reduction method used in many applications to reconstruct the ideal or simplified behavior of a system and its governing equations by taking a complicated dataset and reducing its dimensionality and redundancies. The purpose to reduce linearly correlated variables into linearly uncorrelated variables to simplify the system. The first principal component is the largest as it captures most of the variance in the system. The following principal components capture other uncorrelated variables in the data. Each of the principal component is orthogonal and uncorrelated to the previous one. PCA generally has two steps, first is to normalize the data and the second is to do SVD or do an eigenvalue decomposition of the covariance matrix. For this lab, the SVD method will used, which will work by diagonalizing the covariance matrix.

**SVD**

The SVD is a method that produces the principal components of a normalized dataset. It reduces a matrix to its basic constituent parts to simplify further calculations. The decomposition takes the form:

$$A = U\Sigma V^* \qquad (2)$$

Where A is the original matrix and $U \in \mathbb{C}^{m\times m}$ is unitary, $V \in \mathbb{C}^{n\times n}$ is unitary, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. The *m* denotes the number of measurements while the *n* denotes the number of data points. The matrixes work such that V stores information about the shape of the original matrix, Σ stretches the data with its principal semi-axes, and U rotates the data to its axis. This method give unique singular values, $\sigma_j$, to display variance within the dataset after reducing the dimensions. To compute the SVD, this is the process used:

$$\begin{aligned} A^TA &= (U\Sigma V^*)^T(U\Sigma V^*) \\ &= V\Sigma^2V^* \end{aligned} \qquad (3)$$

And:

$$\begin{aligned} AA^T &= (U\Sigma V^*)(U\Sigma V^*)^T \\ &= U\Sigma^2U^* \end{aligned} \qquad (4)$$

Then, multiplying the right-hand side by V and U, produces two eigenvalue problems.

$$A^TAV = V\Sigma^2 \qquad (5)$$
$$AA^TU = U\Sigma^2 \qquad (6)$$

The square root of the eigenvalues of the equations produces the singular values, $\sigma_j$. Normalized eigenvectors for these equations give the basis vectors for U and V. MATLAB easily does the calculations with *svd* where U, S, and V correspond to U, Σ, and V, respectively.

**Linear Discriminant Analysis (LDA)**
After the dataset is represented by principal components post-SVD or by proper orthogonal mode decomposition, they are classified into limited features. Then, LDA uses statistical information for decision making. LDA has two main underlying concepts: finding a suitable projection that maximizes the distance in samples between different class while minimizing the distance in samples within the same class. Then to construct a projection, seen in equation 7.

$$w = w_{max} \frac{w^T S_B w}{w^T S_W w} \qquad (7)$$

$$S_B = (\mu_{2\_} \mu_1)(\mu_{2\_} \mu_1)^T \qquad (8)$$

$$S_W = \sum_{j=1}^{2} \sum_x (x\_ \mu_j)(x\_ \mu_j)^T \qquad (9)$$

Where $S_B$ denotes the scatter matrix for between class data and is defined by equation 8 and $S_W$ denotes the scatter matrix for within class data and is defined by equation 9. These variables both measure the variance in the data sets and as well as the variance in the difference between means.

$$S_B w = \lambda S_W w \qquad (10)$$

The solution can be found by solving equation 10, where the maximum eigenvalue $\lambda$ and its respective eigenvector show the quantity of interest and the projection basis.

## III. Algorithm Implementation and Development

To easily upload the songs into MATLAB, a function with a for loop was created to easily convert the filename into an *audioread* file via *sprintf*. Then the 5 second clips were *downsample*-d to get every fourth data point to decrease the computational time in a conservative manner. Each song was then concatenated into a matrix. Then the matrix was inputted into a spectrogram function with a Gabor Transform filter. Since we down-sampled each song to get every fourth point, the Fourier mode we chose was the quotient of 44100, the original sampling frequency, and 4. The L was 5 because we chose 5 second audio clips. With this information, we discretized the time and frequency domain, which is then shifted. Once the matrix, still in the time domain, is inputted into the function, a Gaussian filter is applied with a window of 50 and a $\tau$, translation parameter, of 0.1 is applied. As the shifting window centers around different t values, the Gaussian filter with the tslide component is multiplied element-wise to the original signal to only capture the desired portion of the signal. That section of time is taken, Fourier transformed, taken the absolute value of to convert the complex components into real numbers, and fftshift-ed to switch the two halves of the frequency domain. The for-loop creates a matrix with time and frequency data in it. Then, the spectrogram is converted into a row vector and then transposed into column vectors.

These matrices along with a defined variable called feature is inputted into a trainer function. This feature variable entails only a limited number of principal components. In the case of this program, the feature was chosen to be 20, which means the first 20 singular values were used. The 3 matrices are concatenated and the *svd* decomposition is taken. The singular value matrix is multiplied by the transpose of V to find projections and then new variables of each type of song/artist/genre is generated by giving strength of the projections via U. Then equations 8 and 9 were used to find intra and intra-class scatter matrices. Using the *eig* and *max* function, the maximum eigenvalue is found. This new V matrix, associated with the max eigenvalue, contains the new basis for projecting the training dataset with reduced-dimensions approximations of each sample against the other. To accomplish this, the transpose of the new basis vector, *w*, is multiplied with the inputted dataset. The next challenge was to set up a threshold. The average of each of the 3 categorizes were *sort*-ed and given a value of low, medium, and high. Then the index of this was to identify the exact projected result to keep track of the location. Thresholds were found to separate the low and medium projections and the medium and high projections.

Once the training set set-up is done, the test set, a matrix of 30 songs with 10 each from the 3 categories, is converted into a spectrogram and then it undergoes a SVD projection and LDA projection as described by the previous paragraph. A matrix of hidden labels is created to identify what song belongs to what category; in this case, 0, 1, and 2 were assigned to each category. Then manually for values below the first threshold, between the two thresholds, and above the second threshold, a value of 0, 1, or 2 is assigned depending on where the labels should go. Finally, a for-loop is created to see the success rate of a hidden label matching the correct location of where the new testing set data point should go. A histogram is generated along with the W projections.

## IV. Computational Results

**Test Case 1 (Band classification):** The three bands/artists used were Flume, Nat King Cole, and Hall & Oats. The success rate of correctly identifying the test set to the correct artist was 67%. As seen by Figure 01, Flume and Hall & Oats had a pretty big overlap compared to Nat King Cole.

**Test Case 2 (KPOP classification):** Within the same genre of upbeat KPOP music, Girls Generation, Big Bang, and BTS were used. Due to the signature of sings in the same genre, it was more difficult to correctly match the test data to the artist as the success rate was a 33%. As seen in Figure 02, there is a strong overlap between all the artists.

**Test Case 3** (**Genre classification)**: The three genres used were rap, classical, and punk. The success rate of identifying the song to the correct genre was 34%. As seen in Figure 03, classical music had no overlap with rap and punk while those latter two had a significant overlap. This makes sense because of the general frequency of rap/punk music.

All in all, the music classifying algorithm was successful at identifying the different bands if they were different enough. The KPOP and genre classification had half the accuracy of the band classification algorithm.

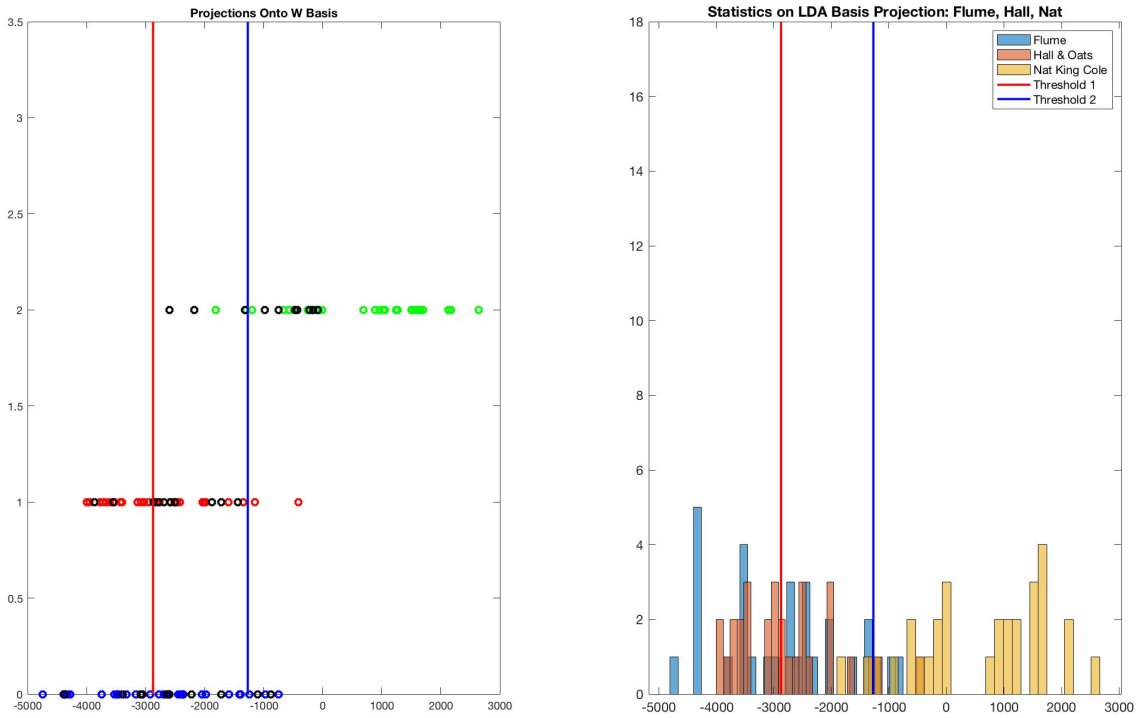


**Figure 01.** Test Case 1- LDA projection onto W basis and a histogram to analyze the LDA basis projections for Flume, Hall & Oats, and Nat King Cole. The black dots for the plot on the left shows projected test data.

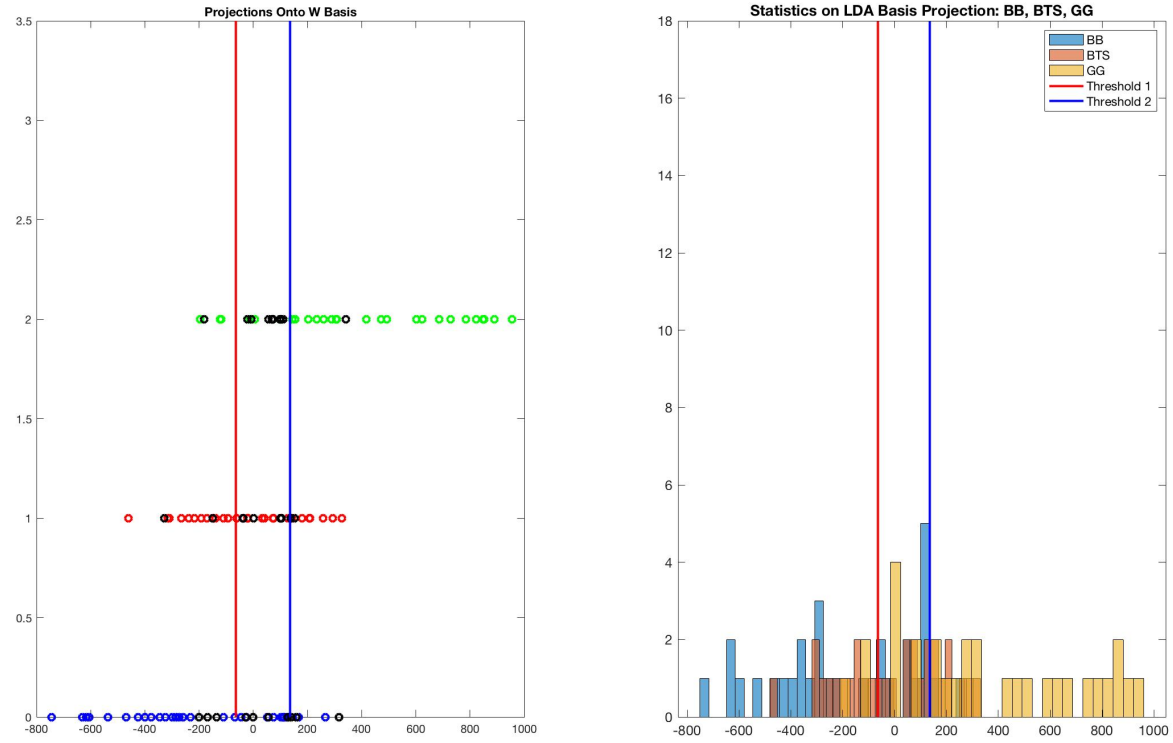


**Figure 02.** Test Case 2- LDA projection onto W basis and a histogram to analyze the LDA basis projections for Girls Generation, Big Bang, and BTS. The black dots for the plot on the left shows projected test data.

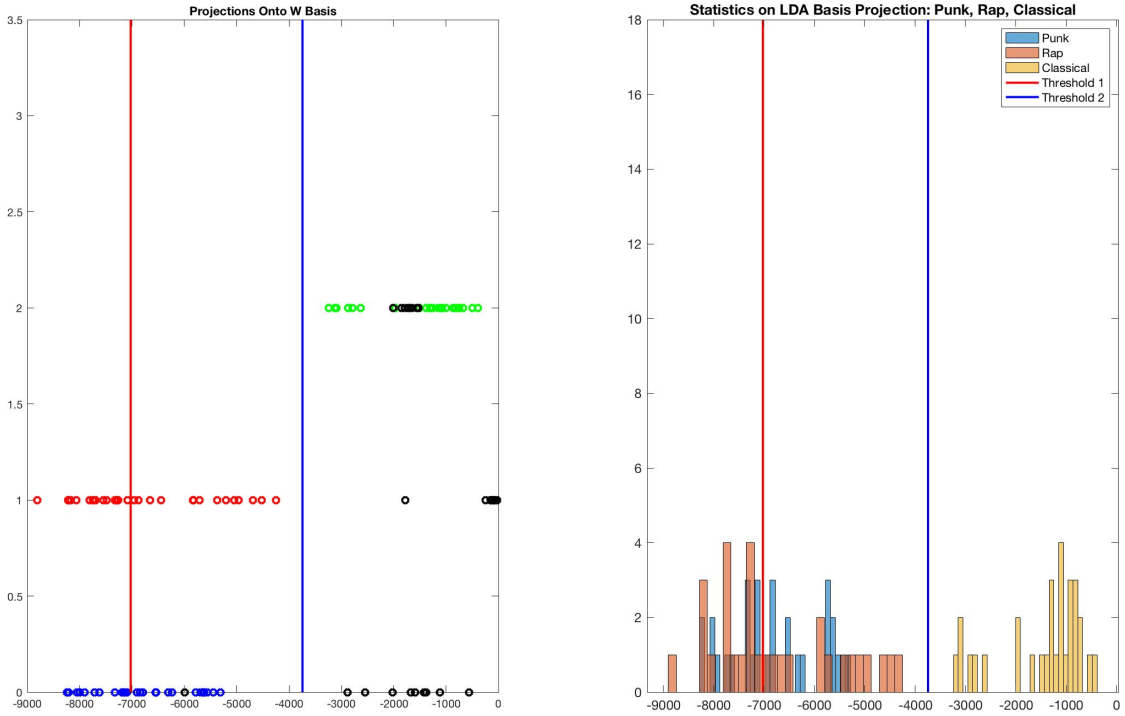


**Figure 03.** Test Case 3- LDA projection onto W basis and a histogram to analyze the LDA basis projections rap, classical, and punk. The black dots for the plot on the left shows projected test data.

# V. Summary and Conclusion

SVD and LDA projections can effectively build dimensionality-reduced machine learning algorithms. In the case of this assignment, the highest accuracy was for the band classification. This makes sense because the artists all sound very different and have different genres. It also makes sense that the genre classification success rate was low because punk and rap have similar tempos. To increase the success rate, more samples need to be used to train the algorithm, using longer than 5 seconds per song would help, and increasing the number of features for projection.

# Appendix A. MATLAB functions used and brief implementation explanation

*svd(x)* – does a singular value decomposition of a given matrix into terms, S, U, V
*eig(x)* – returns eigenvalue of a given matrix
*max(x)* – returns the maximum value in given array
*fftshift(x)* - switches the left and right side of the frequency plot for accurate representation
*fft(x)* - performs the Fast Fourier transform from time to frequency domain
*sort(x)* – organize values in the array in ascending order
*sprint(x)* – returns text as a string, used in this case to easily take music files as input
*downsample(x)* – down-samples the audio vector by specified amount

# Appendix B. MATLAB Codes

## Test case 1.

```
% AMATH 482 Homework 4
% Lahari Gorantla
clear all; close all; clc;

% Test Case 1
% Choose 3 different bands of different genres

%% Uploading the music

num_song = 30;
[flume] = getSong('flume', num_song);
[hall] = getSong('hall', num_song);
[nat] = getSong('nat', num_song);

%% Applying the Spectrogram with a Gabor function

ds = 4; %downsampling
L = 5; % 5 seconds for each audio clip
fs = 44100; fs_d = 44100/ds; %frequency electronic music was sampled at
n = fs_d * L; % number of Fourier modes
t2=linspace(0,L,n+1); t=t2(1:n); % time domain discretization
k=(1/L)*[0:(n-1)/2 (-n+1)/2:-1]; ks=fftshift(k); % freq domain

%obtaining spectro data through a function
[Sgt_flume] = mySpectro(flume, 50, 0.1, L, n, num_song, t);
[Sgt_hall] = mySpectro(hall, 50, 0.1, L, n, num_song, t);
[Sgt_nat] = mySpectro(nat, 50, 0.1, L, n, num_song, t);

%% Taking the SVD

[U,S,V] = svd([Sgt_flume, Sgt_hall, Sgt_nat],'econ');
%% Plotting the Singular Values

figure(1);
semilogy(diag(S),'ko','Linewidth',2);
title('SVD Decomposition');
xlabel('Singular Values');
ylabel('Energy');

%% LDA!

feature = 20;
[U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(Sgt_flume,Sgt_hall,Sgt_nat,feature);

%% Plotting

figure(1);
semilogy(diag(S),'ko','Linewidth',2)
title('SVD Decomposition: Flume, Hall & Oats, Nat King Cole');
xlabel('Singular Values');
ylabel('Energy');

%plot dog/cat projections onto w
figure(2); subplot(2,1,1);
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
title('Projections onto w basis');
ylim([0 2.5])

%making a histogram of overlap with thresholds
subplot(2,1,2);
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: Flume, Hall, Nat')
set(gca,'Fontsize',12)
legend('Flume', 'Hall & Oats','Nat King Cole','Threshold 1','Threshold 2')
```

```
%% Classifying Code

[testset] = getSong('test1', 30);
[test_spect] = mySpectro(testset, 50, 0.1, L, n, num_song, t);

TestNum = 30;
TestMat = U'*test_spect;  % PCA projection
pval = w'*TestMat;  % LDA projection

hiddenlabels = zeros(30,1);
hiddenlabels(1:10) = 0;
hiddenlabels(11:20) = 1;
hiddenlabels(21:30) = 2;

label = zeros(30,1);
for i = 1:30

    if pval(1,i) > threshold_1 && pval(1,i) < threshold_2
        label(i) = 1;

    elseif pval(1,i) > threshold_1 && pval(1,i) > threshold_2
        label(i) = 2;

    else
        label(i) = 0;

    end
end

% add the success rate
count = 0;

for i = 1:30
    if hiddenlabels(i) == label(i)
        count = count + 1;
    end
end

sucRate = count/TestNum;
disp(sucRate);

%%
set(0, 'DefaultLineLineWidth', 2);

subplot(1,2,1)
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
plot(pval(1:10), zeros(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(11:20), 1.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(21:30),2.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2); hold on;
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2); hold on;
%legend('BB','BTS','GG')
title('Projections Onto W Basis');
ylim([0 3.5])

subplot(1,2,2);
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: Flume, Hall, Nat')
set(gca,'Fontsize',12)
legend('Flume', 'Hall & Oats','Nat King Cole','Threshold 1','Threshold 2')



%% Uploading music function
% goal: build a function
% name of the song and the number associated with it
function [music_array] = getSong(song_name,num_song)
    for i = 1:num_song
        file_name = sprintf('%s_%d.wav', song_name, i);
        [y, fs] = audioread(file_name);
        ds_song = downsample(y,4);
```

```
        music_array(i,:) = ds_song(:,1);
    end
end

%% My Spectrogram function

function [sg] = mySpectro(v, width, tau, L, n, num_song, t)
    a = width; tslide=0:tau:L;
    Spec_g = zeros(length(tslide),floor(n)); % store filtered frequency data
    sg = []; % store spectrogram of all 30 songs a row vectors
    for song = 1:num_song
        for j=1:length(tslide)
            % Gabor filter function / window
            g = exp(-a*(t-tslide(j)).^2);
            filtdat = g.*v(song,:); % apply filter to signal (mutiplication in time domain)
            sgt = fft(filtdat);
            Spec_g(j,:) = fftshift(abs(sgt)); % We don't want to scale it
        end

        row_vec = [];
        for i = 1:length(tslide)
            row_vec = [row_vec Spec_g(i, :)];
        end
        sg =[sg; row_vec];
    end
    sg = sg';
end

%% LDA

function [U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(set1,set2,set3,feature);

    % number of columns
    n1 = size(set1,2); n2 = size(set2,2); n3 = size(set3,2);

    [U,S,V] = svd([set1,set2,set3],'econ');

    proj = S*V'; % projection onto principal components
    U = U(:,1:feature);
    proj1 = proj(1:feature,1:n1);
    proj2 = proj(1:feature,n1+1:n1+n2);
    proj3 = proj(1:feature,n1+n2+1:n1+n2+n3);

    mean1 = mean(proj1,2);
    mean2 = mean(proj2,2);
    mean3 = mean(proj3,2);

    Sw = 0; % within class variances

    for k=1:n1
        Sw = Sw + (proj1(:,k)-mean1)*(proj1(:,k)-mean1)';
    end
    for k=1:n2
        Sw = Sw + (proj2(:,k)-mean2)*(proj2(:,k)-mean2)';
    end
    for k=1:n3
        Sw = Sw + (proj3(:,k)-mean3)*(proj3(:,k)-mean3)';
    end

    % between class variance
    bigMean = (mean1+mean2+mean3)/3;
    Sb1 = (mean1-bigMean)*(mean1-bigMean)';
    Sb2 = (mean2-bigMean)*(mean2-bigMean)';
    Sb3 = (mean3-bigMean)*(mean3-bigMean)';
    Sb = (Sb1 + Sb2 + Sb3)/3;

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

    v_1 = w'*proj1;
    v_2 = w'*proj2;
    v_3 = w'*proj3;

    result = [v_1;v_2;v_3];

    sort_mean_1 = mean(v_1);
    sort_mean_2 = mean(v_2);
```

```
    sort_mean_3 = mean(v_3);
    [~,I] = sort([sort_mean_1, sort_mean_2, sort_mean_3]);
    low_vals = I(1);
    med_vals = I(2);
    high_vals = I(3);

    low_p = result(low_vals,:);
    low_p = sort(low_p);
    med_p = result(med_vals,:);
    med_p = sort(med_p);
    high_p = result(high_vals,:);
    high_p = sort(high_p);

    t1 = length(low_p);
    t2 = 1;
    while low_p(t1)>med_p(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold_1 = (low_p(t1)+med_p(t2))/2;

    t2 = length(med_p);
    t3 = 1;
    while med_p(t2)>high_p(t3)
        t2 = t2-1;
        t3 = t3+1;
    end
    threshold_2 = (med_p(t2)+high_p(t3))/2;
end
```

## Test case 2.

```
clear all; close all; clc;

% Test Case 2
% Choose 3 different bands of different genres

%% Uploading the music

num_song = 30;
[bb] = getSong('bb', num_song);
[bts] = getSong('bts', num_song);
[gg] = getSong('gg', num_song);

%% Applying the Spectrogram with a Gabor function

ds = 4; %downsampling
L = 5; % 5 seconds for each audio clip
fs = 44100; fs_d = 44100/ds; %frequency electronic music was sampled at
n = fs_d * L; % number of Fourier modes
t2=linspace(0,L,n+1); t=t2(1:n); % time domain discretization
k=(1/L)*[0:(n-1)/2 (-n+1)/2:-1]; ks=fftshift(k); % freq domain

%obtaining spectro data through a function
[Sgt_bb] = mySpectro(bb, 50, 0.1, L, n, num_song, t);
[Sgt_bts] = mySpectro(bts, 50, 0.1, L, n, num_song, t);
[Sgt_gg] = mySpectro(gg, 50, 0.1, L, n, num_song, t);

%% Taking the SVD

[U,S,V] = svd([Sgt_bb, Sgt_bts, Sgt_gg],'econ');

%% Plotting the Singular Values

figure(1);
semilogy(diag(S),'ko','Linewidth',2);
title('SVD Decomposition');
xlabel('Singular Values');
ylabel('Energy');
%% LDA!

feature = 20;
[U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(Sgt_bb,Sgt_bts,Sgt_gg,feature);

%% Plotting

figure(1);
```

```
semilogy(diag(S),'ko','Linewidth',2)
title('SVD Decomposition: Big Bang, BTS, Girls Generation');
xlabel('Singular Values');
ylabel('Energy');

%plot dog/cat projections onto w
figure(2); subplot(2,1,1);
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
title('Projections onto w basis');
ylim([0 2.5])

%making a histogram of overlap with thresholds
subplot(2,1,2);
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: BB, BTS, GG')
set(gca,'Fontsize',12)
legend('BB', 'BTS','GG','Threshold 1','Threshold 2')

%% Classifying Code

[testset] = getSong('test2', 30);
[test_spect] = mySpectro(testset, 50, 0.1, L, n, num_song, t);

TestNum = 30;
TestMat = U'*test_spect;  % PCA projection
pval = w'*TestMat;  % LDA projection

hiddenlabels = zeros(30,1);
hiddenlabels(1:10) = 0;
hiddenlabels(11:20) = 1;
hiddenlabels(21:30) = 2;

label = zeros(30,1);
for i = 1:30

    if pval(1,i) > threshold_1 && pval(1,i) < threshold_2
        label(i) = 1;

    elseif pval(1,i) > threshold_1 && pval(1,i) > threshold_2
        label(i) = 2;

    else
        label(i) = 0;

    end
end

% add the success rate
count = 0;

for i = 1:30
    if hiddenlabels(i) == label(i)
        count = count + 1;
    end
end

sucRate = count/TestNum;
disp(sucRate);
%%

figure();
set(0, 'DefaultLineLineWidth', 2);
subplot(1,2,1)
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
plot(pval(1:10), zeros(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(11:20), 1.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(21:30),2.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
```

```
%legend('BB','BTS','GG')
title('Projections Onto W Basis');
ylim([0 3.5])

subplot(1,2,2)
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: BB, BTS, GG')
set(gca,'Fontsize',12)
legend('BB', 'BTS','GG','Threshold 1','Threshold 2')

%% Uploading music function
% goal: build a function
% name of the song and the number associated with it
function [music_array] = getSong(song_name,num_song)
    for i = 1:num_song
        file_name = sprintf('%s_%d.wav', song_name, i);
        [y, fs] = audioread(file_name);
        ds_song = downsample(y,4);
        music_array(i,:) = ds_song(:,1);
    end
end

%% My Spectrogram function

function [sg] = mySpectro(v, width, tau, L, n, num_song, t)
    a = width; tslide=0:tau:L;
    Spec_g = zeros(length(tslide),floor(n)); % store filtered frequency data
    sg = []; % store spectrogram of all 30 songs a row vectors
    for song = 1:num_song
        for j=1:length(tslide)
            % Gabor filter function / window
            g = exp(-a*(t-tslide(j)).^2);
            filtdat = g.*v(song,:); % apply filter to signal (mutiplication in time domain)
            sgt = fft(filtdat);
            Spec_g(j,:) = fftshift(abs(sgt)); % We don't want to scale it
        end

        row_vec = [];
        for i = 1:length(tslide)
            row_vec = [row_vec Spec_g(i, :)];
        end
        sg =[sg; row_vec];
    end
    sg = sg';
end

%% LDA

function [U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(set1,set2,set3,feature);

    % number of columns
    n1 = size(set1,2); n2 = size(set2,2); n3 = size(set3,2);

    [U,S,V] = svd([set1,set2,set3],'econ');

    proj = S*V'; % projection onto principal components
    U = U(:,1:feature);
    proj1 = proj(1:feature,1:n1);
    proj2 = proj(1:feature,n1+1:n1+n2);
    proj3 = proj(1:feature,n1+n2+1:n1+n2+n3);

    mean1 = mean(proj1,2);
    mean2 = mean(proj2,2);
    mean3 = mean(proj3,2);

    Sw = 0; % within class variances

    for k=1:n1
        Sw = Sw + (proj1(:,k)-mean1)*(proj1(:,k)-mean1)';
    end
    for k=1:n2
        Sw = Sw + (proj2(:,k)-mean2)*(proj2(:,k)-mean2)';
    end
```

```
    for k=1:n3
        Sw = Sw + (proj3(:,k)-mean3)*(proj3(:,k)-mean3)';
    end

    % between class variance
    bigMean = (mean1+mean2+mean3)/3;
    Sb1 = (mean1-bigMean)*(mean1-bigMean)';
    Sb2 = (mean2-bigMean)*(mean2-bigMean)';
    Sb3 = (mean3-bigMean)*(mean3-bigMean)';
    Sb = (Sb1 + Sb2 + Sb3)/3;

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

    v_1 = w'*proj1;
    v_2 = w'*proj2;
    v_3 = w'*proj3;

    result = [v_1;v_2;v_3];

    sort_mean_1 = mean(v_1);
    sort_mean_2 = mean(v_2);
    sort_mean_3 = mean(v_3);
    [~,I] = sort([sort_mean_1, sort_mean_2, sort_mean_3]);
    low_vals = I(1);
    med_vals = I(2);
    high_vals = I(3);

    low_p = result(low_vals,:);
    low_p = sort(low_p);
    med_p = result(med_vals,:);
    med_p = sort(med_p);
    high_p = result(high_vals,:);
    high_p = sort(high_p);

    t1 = length(low_p);
    t2 = 1;
    while low_p(t1)>med_p(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold_1 = (low_p(t1)+med_p(t2))/2;

    t2 = length(med_p);
    t3 = 1;
    while med_p(t2)>high_p(t3)
        t2 = t2-1;
        t3 = t3+1;
    end
    threshold_2 = (med_p(t2)+high_p(t3))/2;
end
```

## Test case 3.

```
% AMATH 482 Homework 4
% Lahari Gorantla
clear all; close all; clc;

% Test Case 2
% Choose 3 different bands of different genres

%% Uploading the music

num_song = 30;
[punk] = getSong('punk', num_song);
[classical] = getSong('classical', num_song);
[rap] = getSong('rap', num_song);

%% Applying the Spectrogram with a Gabor function

ds = 4; %downsampling
L = 5; % 5 seconds for each audio clip
fs = 44100; fs_d = 44100/ds; %frequency electronic music was sampled at
n = fs_d * L; % number of Fourier modes
t2=linspace(0,L,n+1); t=t2(1:n); % time domain discretization
k=(1/L)*[0:(n-1)/2 (-n+1)/2:-1]; ks=fftshift(k); % freq domain
```

```
%obtaining spectro data through a function
[Sgt_punk] = mySpectro(punk, 50, 0.1, L, n, num_song, t);
[Sgt_class] = mySpectro(classical, 50, 0.1, L, n, num_song, t);
[Sgt_rap] = mySpectro(rap, 50, 0.1, L, n, num_song, t);

%% Taking the SVD

[U,S,V] = svd([Sgt_punk, Sgt_class, Sgt_rap],'econ');

%% Plotting the Singular Values

figure(1);
semilogy(diag(S),'ko','Linewidth',2);
title('SVD Decomposition');
xlabel('Singular Values');
ylabel('Energy');
%% LDA!

feature = 20;
[U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(Sgt_punk,Sgt_class,Sgt_rap,feature);

%% Plotting

figure(1);
semilogy(diag(S),'ko','Linewidth',2)
title('SVD Decomposition: punk, classical, rap');
xlabel('Singular Values');
ylabel('Energy');

%plot dog/cat projections onto w
figure(2); subplot(2,1,1);
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
title('Projections onto w basis');
ylim([0 2.5])

%making a histogram of overlap with thresholds
subplot(2,1,2);
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: punk, classical, rap')
set(gca,'Fontsize',12)
legend('punk', 'classical', rap,'Threshold 1','Threshold 2')

%% Classifying Code

[testset] = getSong('test3', 30);
[test_spect] = mySpectro(testset, 50, 0.1, L, n, num_song, t);

TestNum = 30;
TestMat = U'*test_spect;  % PCA projection
pval = w'*TestMat;  % LDA projection

hiddenlabels = zeros(30,1);
hiddenlabels(1:10) = 0;
hiddenlabels(11:20) = 1;
hiddenlabels(21:30) = 2;

label = zeros(30,1);
for i = 1:30

    if pval(1,i) > threshold_1 && pval(1,i) < threshold_2
        label(i) = 1;

    elseif pval(1,i) > threshold_1 && pval(1,i) > threshold_2
        label(i) = 2;

    else
        label(i) = 0;

    end
end
```

```
% add the success rate
count = 0;

for i = 1:30
    if hiddenlabels(i) == label(i)
        count = count + 1;
    end
end

sucRate = count/TestNum;
disp(sucRate);
%%

figure();
set(0, 'DefaultLineLineWidth', 2);
subplot(1,2,1)
plot(low_p,zeros(length(low_p)),'ob','Linewidth',2); hold on
plot(med_p,1.*ones(length(med_p)),'or','Linewidth',2); hold on
plot(high_p, 2.*ones(length(high_p)),'og','Linewidth',2); hold on
plot(pval(1:10), zeros(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(11:20), 1.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot(pval(21:30),2.*ones(length(1:10)),'ok','Linewidth',2); hold on;
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
%legend('BB','BTS','GG')
title('Projections Onto W Basis');
ylim([0 3.5])

subplot(1,2,2)
histogram(low_p,length(low_p))
hold on
histogram(med_p,length(med_p))
histogram(high_p,length(high_p))
plot([threshold_1 threshold_1],[0 18],'r','LineWidth',2)
plot([threshold_2 threshold_2],[0 18],'b','LineWidth',2)
title('Statistics on LDA Basis Projection: Punk, Rap, Classical')
set(gca,'Fontsize',12)
legend('Punk', 'Rap','Classical','Threshold 1','Threshold 2')




%% Uploading music function
% goal: build a function
% name of the song and the number associated with it
function [music_array] = getSong(song_name,num_song)
    for i = 1:num_song
        file_name = sprintf('%s_%d.wav', song_name, i);
        [y, fs] = audioread(file_name);
        ds_song = downsample(y,4);
        music_array(i,:) = ds_song(:,1);
    end
end

%% My Spectrogram function

function [sg] = mySpectro(v, width, tau, L, n, num_song, t)
    a = width; tslide=0:tau:L;
    Spec_g = zeros(length(tslide),floor(n)); % store filtered frequency data
    sg = []; % store spectrogram of all 30 songs a row vectors
    for song = 1:num_song
        for j=1:length(tslide)
            % Gabor filter function / window
            g = exp(-a*(t-tslide(j)).^2);
            filtdat = g.*v(song,:); % apply filter to signal (mutiplication in time domain)
            sgt = fft(filtdat);
            Spec_g(j,:) = fftshift(abs(sgt)); % We don't want to scale it
        end

        row_vec = [];
        for i = 1:length(tslide)
            row_vec = [row_vec Spec_g(i, :)];
        end
        sg =[sg; row_vec];
    end
    sg = sg';
end
```

```
%% LDA

function [U,S,V,threshold_1, threshold_2,w,low_p,med_p,high_p] = dc_trainer(set1,set2,set3,feature);

    % number of columns
    n1 = size(set1,2); n2 = size(set2,2); n3 = size(set3,2);

    [U,S,V] = svd([set1,set2,set3],'econ');

    proj = S*V'; % projection onto principal components
    U = U(:,1:feature);
    proj1 = proj(1:feature,1:n1);
    proj2 = proj(1:feature,n1+1:n1+n2);
    proj3 = proj(1:feature,n1+n2+1:n1+n2+n3);

    mean1 = mean(proj1,2);
    mean2 = mean(proj2,2);
    mean3 = mean(proj3,2);

    Sw = 0; % within class variances

    for k=1:n1
        Sw = Sw + (proj1(:,k)-mean1)*(proj1(:,k)-mean1)';
    end
    for k=1:n2
        Sw = Sw + (proj2(:,k)-mean2)*(proj2(:,k)-mean2)';
    end
    for k=1:n3
        Sw = Sw + (proj3(:,k)-mean3)*(proj3(:,k)-mean3)';
    end

    % between class variance
    bigMean = (mean1+mean2+mean3)/3;
    Sb1 = (mean1-bigMean)*(mean1-bigMean)';
    Sb2 = (mean2-bigMean)*(mean2-bigMean)';
    Sb3 = (mean3-bigMean)*(mean3-bigMean)';
    Sb = (Sb1 + Sb2 + Sb3)/3;

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

    v_1 = w'*proj1;
    v_2 = w'*proj2;
    v_3 = w'*proj3;

    result = [v_1;v_2;v_3];

    sort_mean_1 = mean(v_1);
    sort_mean_2 = mean(v_2);
    sort_mean_3 = mean(v_3);
    [~,I] = sort([sort_mean_1, sort_mean_2, sort_mean_3]);
    low_vals = I(1);
    med_vals = I(2);
    high_vals = I(3);

    low_p = result(low_vals,:);
    low_p = sort(low_p);
    med_p = result(med_vals,:);
    med_p = sort(med_p);
    high_p = result(high_vals,:);
    high_p = sort(high_p);

    t1 = length(low_p);
    t2 = 1;
    while low_p(t1)>med_p(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold_1 = (low_p(t1)+med_p(t2))/2;

    t2 = length(med_p);
    t3 = 1;
    while med_p(t2)>high_p(t3)
        t2 = t2-1;
        t3 = t3+1;
    end
```

```
    threshold_2 = (med_p(t2)+high_p(t3))/2;
end
```