Lab 3

Clint Olsen

ECEN 4532: DSP Lab

March 6, 2018

University of Colorado
Boulder

Contents

Introduction

This goal of this lab is to manipulate the results of high level audio signal descriptors generated in the previous lab to begin constructing a more sophisticated classification algorithms to perform on various genres of audio tracks. This will be the first step in constructing a genre classifier, which is the first step to identifying music at a much finer level, such as single song identification performed by applications such as Shazam and SoundHound. The basis for this lab will be determining the relative distances between all of the tracks and implementing the k-nearest neighbors algorithm on this data to begin classifying songs into particular genres. From here, a more extensive experiment will be conducted using Cross Validation to see how accurate the classifier is with this algorithm. The analysis will conclude by attempting to further improve the performance of the classification using more sophisticated algorithms. Throughout the lab, MFCC coefficients will be used as the main characteristic to describe all of the audio signals and all audio tracks will be 2 minutes in length, or less if the entire track is less than 2 minutes.

Assignment 1

In order to begin the classification analysis, the main structure that will be used to classify the tracks is the distance matrix. The computation involved with this calculation is quite extensive so the first step is the condense the 40 MFCC coefficients to 12, as specified by the report. The described method to do this is to combine frequency responses that are in the middle of the human auditory range into individual MFCC coefficients, where many MFCC values near the low and high ends of the spectrum are grouped into 1 MFCC value, because these frequencies are less common and less responsive within the human auditory response. In order to find the distance matrix, each song will be compared with each song to find the distance between them. This distance is found by finding the distance between the Gaussian Distribution of each track, resulting in the Kullback-Leibler Divergence, which describes the distance between all of the tracks. The MATLAB code for this computation is found in the Assignment 1 Appendix

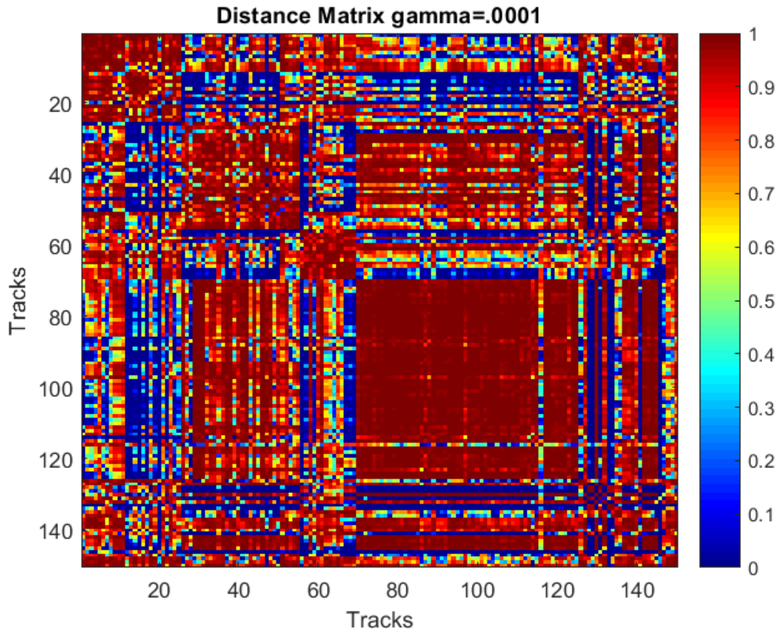
$$KL(G^s, G^{s'}) = \frac{1}{2}\mathrm{tr}(\Sigma_{s'}^{-1}\Sigma_s + \Sigma_s^{-1}\Sigma_{s'}) - 12 + \frac{1}{2}(\mu_s - \mu_{s'})^T\left(\Sigma_{s'}^{-1} + \Sigma_s^{-1}\right)(\mu_s - \mu_{s'}) \quad (5)$$

Assignment 2

Extending from the Kullback-Leibler divergence, a rescaled version of this matrix can be found, which is the distance matrix itself described by the following equation:

$$d(s, s') = e^{-\gamma KL(G^s, G^{s'})}$$

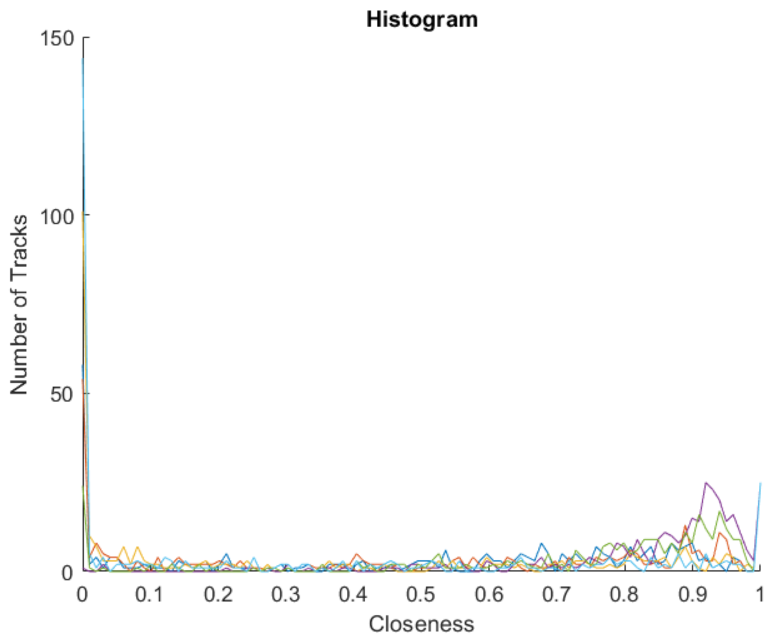
Intuitively, this scales the distances by a factor of γ within the matrix and flips the magnitude such that distances with a large value correspond to tracks that are "close" in terms of the power carried in the MFCC coefficients. Below is the distance matrix with a γ value of .0001.



This plot gives several intuitive notes regarding these tracks. Due to the fact that the tracks are analyzed genre by genre, each 25x25 block along the diagonal corresponds to songs that are in the same genre. The plot shows that these blocks along the diagonal of the matrix have a high intensity, which means they are "close" together. This makes sense because the tracks are of the same genre which in general will have a consistent loudness, timbre and overall energy. A side note is that the matrix is also symmetric, due to the fact the distance from song A to song B is the same as the distance from song B to song A within the matrix. This plot can also be used to analyze the similarities or differences between genres. For example, looking at the x indices from 76 to 100 and the y indices from 101 to 125, it can be seen that there is high intensity within this region, which is due to the fact that the x axis here correlates to punk rock music and the y axis correlates to rock music. These genres aurally share many similarities in loundness, rhythm and monotony, so it makes sense these are depicted as "similar". Another example is how the indices that correlate with world tracks, 126 to 150, do not share a very large correlation with any other genre, due to the fact that musically, this genre varies very much, even within itself, with other genres in this classification. Another side note is the fact that this plot is not identical the example provided in the Lecture 6 slides, which is due to the fact that the tracks were read in different order. This does not effect the overall result of the analysis, just shifts the location of the genre blocks slightly within themselves. The code for the distance matrix can be found in the Assignment 2 and Lab 3 Driver Appendix.

Assignment 3

To further exemplify the operation of the distance matrix, this section focuses on creating a histogram of the distances within a particular genre. This is done by summing the amount of times a particular distance is seen between two tracks of the same genre. The below plot is shown with a γ value of .002



The plot shows the overlapped histograms of all 6 genres. It can be seen that there is are many peaks near 1 on the x-axis, which is expected because tracks within the same genre should ideally be "close" in terms of the distance matrix. Another note to make is that the sum of the number of tracks at each intensity level should sum to 25 for each of the 6 plots, because there are that many tracks within each genre. Note that this matrix is also symmetric. See the MATLAB code for generating the histogram in the Assignment 3 and Lab 3 Driver Appendix.

Assignment 4

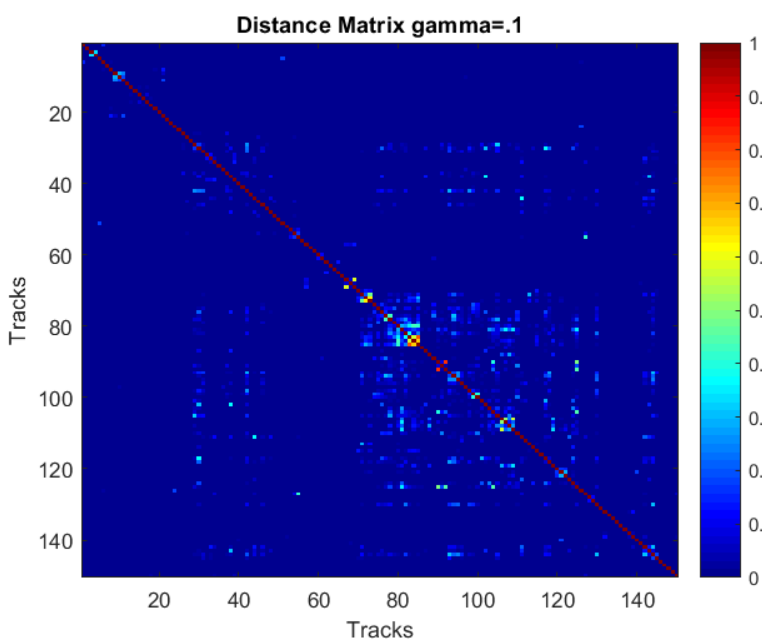
In order to see the distances and closeness between genres numerically, a simple average can be taken of the distance matrix to show the average distance between all of the genres

	Classical	Electronic	Jazz	Punk	Rock	World
Classical	0.505751652	0.085281167	0.19628828	0.043439841	0.030998251	0.231177
Electronic	0.085281167	0.482099549	0.238288615	0.469634785	0.416588434	0.249729
Jazz	0.19628828	0.238288615	0.352720511	0.292059283	0.268442133	0.262414
Punk	0.043439841	0.469634785	0.292059283	0.83540398	0.736858947	0.266992
Rock	0.030998251	0.416588434	0.268442133	0.736858947	0.724362854	0.242932
World	0.23117714	0.249729142	0.262413588	0.266991856	0.242932224	0.29177

This gives a more concise model to the method used above of visually inspecting the distance matrix genre blocks and looking at the intensity, but yields similar results. Recall that numbers close to 1 in the matrix correlate to genres that are "close". Using the same example as in Assignment 2, column 4 row 5 of the table shows that Punk (genre 4) and Rock (genre 5) have a large value indicating closeness, whereas world music tends to have smaller values for each corresponding genre in the table. The code for this section can be found in the Assignment 4 Appendix.

Assignment 5

The variation of the γ parameters greatly changes the visual output of the distance matrix shown in Assignment 2. An interesting note is the fact that the remainder of the lab, including the k-nearest neighbor algorithm does not depend on the parameter γ. In looking at the plots, large values of gamma increase the overall distance between genres. This means that as γ gets larger, it starts reducing the lower intensity parts of the distance matrix. This can be seen from the average distance matrix as well, where the example of punk compared to rock in the Assignment 4 has a smaller distance between the genres at .002, at a large value the distance matrix becomes sparse, indicating greater distance (lower intensity value).



	Classical	Electronic	Jazz	Punk	Rock	World
Classical	0.046013836	6.60E-06	0.00034726	5.05E-05	1.81E-09	0.000478
Electronic	6.60E-06	0.048829473	0.000817758	0.00680792	0.006849858	0.003408
Jazz	0.00034726	0.000817758	0.050303798	0.007272612	0.002144117	0.001794
Punk	5.05E-05	0.00680792	0.007272612	0.086280727	0.021834987	0.005659
Rock	1.81E-09	0.006849858	0.002144117	0.021834987	0.067387971	0.003248
World	0.000478087	0.003407822	0.001794296	0.005659089	0.003248044	0.044886

Assignment 8

To begin formally classifying the tracks into the 6 given genres, the k-nearest neighbors algorithms will be applied to the distance matrix, with a k value of 5. This means that the tracks with the 5 greatest values in the distance matrix row correlating to that particular track will be translated into a genre and from there, the genre out of those 5 with the greatest presence will be used to increment a confusion matrix for that the genre of the track, and the majority genre given by the algorithm. The result shows how many songs in a particular genre actually got mapped back to the correct genre, with a ideal case of all 25s down the diagonal, signaling perfect classification.

	Classical	Electronic	Jazz	Punk	Rock	World
Classical	25	0.00E+00	0	0.00E+00	0.00E+00	0
Electronic	2.00E+00	17	0	3	2	1
Jazz	3	3	15	4	0	0
Punk	1.00E+00	2	2	14	6	0
Rock	1.00E+00	4	1	6	13	0
World	7	3	4	2	2	7

From the confusion matrix above, it can be seen that the algorithm is of course, not perfect. Although, it does show that some genres are easier to classify than others using this particular algorithm. For example, classical, shown in matrix column and row 1, has a perfect 25, meaning that in this scenario, the algorithm was able to classify all of the classical tracks correctly. This shows that the tracks had little variance in terms of their spectral characteristics and energy from track to track. The opposite case is the world genre, row and column 6, has a value of 7 meaning that only 28% of the songs were classified correctly. This is due to the lack of consistency between the world tracks, the avant garde nature of the songs melodically, and the various timbres that comprise each track. The MATLAB code for this algorithm can be found in the Assignment 8 Appendix.

Assignment 9

To further evaluate the performance of this classification, a 5 fold cross validation technique is used. This consists of creating a randomized training and test set for the classification and running it several times. For this test, the tracks in each genre will be randomized into 5 sections of 5 songs each, giving the test set size of 30, 5 songs for each 6 genres. Overall, this will create 5 total test sets that will be iterated through. The 120 songs that are not used in each iteration as a test set will be used as the training set, of the set of tracks to compare the test set with. This experiment will be run 10 times, resulting in 50 confusion matrices. Running the test this many times will give a performance profile of the algorithm as a whole and how accurate it is in general. Below, the mean and standard deviation of the 50 confusion matrices are shown:

Mean	Classical	Electronic	Jazz	Punk	Rock	World
Classical	4.82	0.00E+00	0.06	0.00E+00	0.00E+00	0.12
Electronic	2.60E-01	3.46	0.24	0.5	0.32	0.22
Jazz	0.46	0.52	3.12	0.7	0.04	0.16
Punk	1.00E-01	0.66	0.22	2.96	1.06	0
Rock	2.00E-01	0.54	0.2	1.38	2.64	0.04
World	1.68	0.68	0.72	0.38	0.24	1.3

Std Dev	Classical	Electronic	Jazz	Punk	Rock	World
Classical	0.494871659	0.00E+00	0.303045763	0.00E+00	0.00E+00	0.303046
Electronic	5.36E-01	0.968904283	0.572855362	0.646497629	0.578879988	0.664247
Jazz	0.706818107	0.676425177	1.006914868	0.677630927	0.197948664	0.328261
Punk	3.88E-01	0.702473763	0.451753951	0.958101865	0.769043933	0
Rock	4.04E-01	0.745325569	0.476380914	0.935468888	0.973317491	0.197949
World	1.038837655	0.702473763	0.717421687	0.609114446	0.597955701	1.111168

The result of the cross validation gives a different result upon each run because of the fact the songs are randomized upon runtime. The idea is to see how accurate the algorithm is over many iterations, so for the example shown above of the mean, the accuracy is about 61% which compared with the single run case from Assignment 8 (60.67%) shows that this is an accurate profile for the performance of the algorithm. The standard deviation also gives results on the accuracy of the classification due to the fact that a smaller deviation indicates a more consistent classification of a particular genre, such as the classical genre having a low standard deviation and a large mean, as well as a large deviation of the data indicating inaccurate classification (world tracks). This results have been consistently observed throughout the lab for these genres. The code for this portion is shown in the Assignment 9 Appendix.

Assignment 11

The final piece to this lab is to attempt to improve the classification performance for these tracks with the use of a more sophisticated algorithm. For this example, a Support Vector Machine was used by implementing the MATLAB Machine Learning and Statistics Toolbox. The classes that were used were single row vectors for each track in the training set comprised of the mean and covariance of the distances for that particular track. These classes fed into the machine are just each of the six genres. After the model of the training set is generated, it can be compared with each test set (this is still a 5-fold cross validation) to determine classifications. After testing, running just a single experiment took a great deal of time, so instead of 10 experiments like in the previous Assignments, only 1 was used. The results were actually worse than the other cross validation technique with the k-nearest neighbors algorithm, coming in around 51.33% accuracy. There are many reasons for this. The first is the fact that many tests couldn't be run due to how resource and time intensive the computation was for a Laptop computer, so more test would definitely be necessary. The other consideration is the data used to generate the model; the covariance and mean. For this particular algorithm, this may not be the best choice of data characteristics and perhaps another data set, such as MFCC values or another aural characteristic (or maybe several) would be better suited. The code for this implementation is shown in the Assignment 11 Appendix.

Code Appendix

Lab 3 Driver Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3 Driver Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% File Reading
classicalDir = strcat(pwd,'\data\classical\');
electronicDir = strcat(pwd,'\data\electronic\');
jazzDir = strcat(pwd,'\data\jazz\');
punkDir = strcat(pwd,'\data\punk\');
rockDir = strcat(pwd,'\data\rock\');
worldDir = strcat(pwd,'\data\world\');

classical = dir(classicalDir);
classical = {classical(3:length(classical)).name}';
electronic = dir(electronicDir);
electronic = {electronic(3:length(electronic)).name}';
jazz = dir(jazzDir);
jazz = {jazz(3:length(jazz)).name}';
punk = dir(punkDir);
punk = {punk(3:length(punk)).name}';
rock = dir(rockDir);
rock = {rock(3:length(rock)).name}';
world = dir(worldDir);
world = {world(3:length(world)).name}';

%% Assignment 1: Extract the merged MFCC Coefficients
N = 512; %Frame size
K = N/2 + 1;
Overlap = 256; %Frame Overlap
T = 120; %Song Length (seconds)
Fs = 22050; %Sample Rate
Nb = 40; %Filter banks

%Mean, Cov, and Inv. Cov Matrices
mu_tracks = zeros(12,150);
cov_tracks = zeros(12,12,150);
cov_inv_tracks = zeros(12,12,150);

% Perform the Prosessing on All tracks
for i=1:25

    %%%%%%%%%%%%%%%%%%%%%%%%Process Classical %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(classicalDir,classical(i))));
    classicalTrack = extractTSeconds(T,char(strcat(classicalDir,classical(i))));

    %Determine the track with overlap of 256
    track = zeros(N,floor(length(classicalTrack/Overlap)));
    track = buffer(classicalTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i) = mean(mel2',2);
    cov_tracks(:,:,i) = cov(mel2);
    cov_inv_tracks(:,:,i) = inv(cov_tracks(:,:,i));

    %%%%%%%%%%%%%%%%%%%%%%%%Process Electronic%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(electronicDir,electronic(i))));
    electronicTrack = extractTSeconds(T,char(strcat(electronicDir,electronic(i))));

    %Determine the track with overlap of 256
    track = zeros(N,length(electronicTrack/Overlap));
    track = buffer(electronicTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i+25) = mean(mel2',2);
    cov_tracks(:,:,i+25) = cov(mel2);
    cov_inv_tracks(:,:,i+25) = inv(cov_tracks(:,:,i+25));

    %%%%%%%%%%%%%%%%%%%%%%%%Process Jazz%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(jazzDir,jazz(i))));
    jazzTrack = extractTSeconds(T,char(strcat(jazzDir,jazz(i))));

    %Determine the track with overlap of 256
```

```
    track = zeros(N,length(jazzTrack/Overlap));
    track = buffer(jazzTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i+50) = mean(mel2',2);
    cov_tracks(:,:,i+50) = cov(mel2);
    cov_inv_tracks(:,:,i+50) = inv(cov_tracks(:,:,i+50));

    %%%%%%%%%%%%%%%%%%%%Process Punk%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(punkDir,punk(i))));
    punkTrack = extractTSeconds(T,char(strcat(punkDir,punk(i))));

    %Determine the track with overlap of 256
    track = zeros(N,length(punkTrack/Overlap));
    track = buffer(punkTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i+75) = mean(mel2',2);
    cov_tracks(:,:,i+75) = cov(mel2);
    cov_inv_tracks(:,:,i+75) = inv(cov_tracks(:,:,i+75));

    %%%%%%%%%%%%%%%%%%%%Process Rock%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(rockDir,rock(i))));
    rockTrack = extractTSeconds(T,char(strcat(rockDir,rock(i))));

    %Determine the track with overlap of 256
    track = zeros(N,length(rockTrack/Overlap));
    track = buffer(rockTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i+100) = mean(mel2',2);
    cov_tracks(:,:,i+100) = cov(mel2);
    cov_inv_tracks(:,:,i+100) = inv(cov_tracks(:,:,i+100));

    %%%%%%%%%%%%%%%%%%%%Process World%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Extract 2 minutes (or whole track) from file
    fbank = cochlearFilterbank(N,char(strcat(worldDir,world(i))));
    worldTrack = extractTSeconds(T,char(strcat(worldDir,world(i))));

    %Determine the track with overlap of 256
    track = zeros(N,length(worldTrack/Overlap));
    track = buffer(worldTrack,N,Overlap);
    mel2 = zeros(floor(length(track)),12);
    mel2 = mergedmfcc(track,N,Nb,Overlap,fbank);

    %Calculate mean, covariance, and inverse covariance
    mu_tracks(:,i+125) = mean(mel2',2);
    cov_tracks(:,:,i+125) = cov(mel2);
    cov_inv_tracks(:,:,i+125) = inv(cov_tracks(:,:,i+125));

end

%% Assignment 1: KL Divergence and distance matrix

%Compute the KL divergence
KL = zeros(150,150);
KL = KLD(mu_tracks,cov_tracks,cov_inv_tracks);

%Write the distance matrix to a file
fd = fopen('kld.dat', 'w');
fwrite(fd, KL, 'float64');
fclose(fd);

%% Assignment 2: Plot the distance matrix
%Distance Matrices
KL = zeros(150,150);
d = zeros(150,150);

%Read in KL from file
fd = fopen('kld.dat','r');
KL = fread(fd,[150,150],'float64');
fclose(fd);

%Compute Distance Matrix
d = distance(KL,.1);

%Plot Distance Matrix
colormap('jet')
imagesc(d);
title('Distance Matrix gamma=.1');
xlabel('Tracks');
```

```
ylabel('Tracks');
colorbar();

%% Assignment 3: Distance Histogram
% Calculate histogram
hist = distHist(d);
bincounts = zeros(6,100);
binranges = linspace(0,1,100);

%Plot the histogram from all 6 genres on 1 plot
hold on;
 for k=1:6
    bincounts(k,:) = histc(hist(k,:),binranges);
    bincounts = bincounts/2;
    bincounts(k,100) = bincounts(k,100) + 12.5;
    plot(binranges, bincounts(k,:));
 end
 title('Histogram');
 xlabel('Closeness');
 ylabel('Number of Tracks');
hold off;

%% Assignment 4: Average Distance Matrix
%Calculate avg distance matrix
ADM = avgDistMat(d);

%% Assignment 8: K-Nearest Neighbors

%Calculate the k-nearest neighbors for k=5
k = zeros(6,6);
k = kNeighbors(d,5);

%% Assignment 9: Cross Validation Mean and Std Dev
%Run the cross validation
CV = zeros(6,6,50);
CVMean = zeros(6,6);
CVDev = zeros(6,6);
CV = crossValidation(d);
CVMean = mean(CV,3); %Determine the Mean
CVDev = std(CV,0,3); %Determine Std. Dev
Accuracy = trace(CVMean)/30;

%% Assignment 11: Cross Validation Improvement
%Run the cross validation improved
CVImproved = zeros(6,6,5);
CVMeanImproved = zeros(6,6);
CVDevImproved = zeros(6,6);
CVImproved = improvedClassification(d,mu_tracks,cov_tracks);
CVMeanImproved = mean(CVImproved,3); %Determine the Mean
CVDevImproved = std(CVImproved,0,3); %Determine Std. Dev
AccuracyImproved = trace(CVMeanImproved)/30;
```

Assignment 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Kullback-Leibler Divergence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function KL = KLD(mu,cov,cov_inv)
    %Declare matrix for distances
    KL = zeros(150,150);

    % Loop through all combos of each track (only upper diag)
    for s=1:150
        for sprime=s:150
            %Perform the KL divergence calculation
            KL(s,sprime) = .5*trace(cov_inv(:,:,sprime)*cov(:,:,s)+cov_inv(:,:,s)*cov(:,:,sprime)) ...
                -12+ .5*(mu(:,s)-mu(:,sprime))'*(cov_inv(:,:,sprime)+cov_inv(:,:,s))*(mu(:,s)-mu(:,sprime)
                    );
            KL(sprime,s) = KL(s,sprime);
        end
    end

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Merged MFCC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function mel2 = mergedmfcc(track,N,Nb,Overlap,fbank)
    %Variables for MFCC values
    mfccdB = zeros(floor(length(track)),Nb);
    mfcc = zeros(floor(length(track)),Nb);
    K = N/2 + 1;

```

```
    % Find mfcc coef. for each frame in track
    for n=1:floor(length(track))
        %Extract N samples based on frame number
        xn = track(:,n);

        %Perform the fft and extract the desired components up to K and
        %determine the MFCC coefficients
        Y = abs(fft(xn));
        Xn = Y(1:K);
        [mfccdB(n,:) mfcc(n,:)] = mfccComp(N,Xn,fbank,Nb);
    end

    % New merge mappings
    t = zeros(1,36);
    t(1) = 1;
    t(2) = 2;
    t(3:4) = 3;
    t(5:6) = 4;
    t(7:8) = 5;
    t(9:10) = 6;
    t(11:12) = 7;
    t(13:14) = 8;
    t(15:18) = 9;
    t(19:23) = 10;
    t(24:29) = 11;
    t(30:36) = 12;

    % Merge the 40 MFCC values to 12 values
    mel2 = zeros(floor(length(track)),12);
    for i=1:12
        sum = zeros(floor(length(track)),1);
        index = find(t==i);
        for j=1:length(index)
            sum = sum + mfcc(:,index(j));
        end
        mel2(:,i) = sum;
    end

end
```

Assignment 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Distance Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d = distance(KL, gamma)
    %Declare matrix for distances
    d = zeros(150,150);

    %Calculate Distace Matrix (loop only through upper diag
    d =  exp(-gamma*KL);
end
```

Assignment 3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Distance Histogram
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function hist = distHist(d)
    %Declare the histogram vectors
    hist = zeros(6,625);

    for g=1:6
        for i=1:25
            for j=1:25 %Sum the distances from each genre
                hist(g,((i-1)*25)+ j) = d(i+(25*(g-1)),j+(25*(g-1)));
            end
        end
    end
end
```

Assignment 4

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Average Distance Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```matlab

function avg = avgDistMat(d)
    %Declare Average Distance Matrix
    avg = zeros(6,6);

    %Calculate ADM
    for i=1:6
        for j=1:6
            avg(i,j) = (1/(25^2))*sum(sum(d((((i-1)*25)+1):(((i-1)*25)+25), ...
                (((j-1)*25)+1):(((j-1)*25)+25))));
        end
    end

end
```

Assignment 8

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: K-Nearest Neighbors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function K = kNeighbors(d,k)
    num_in_genre = zeros(1,6);

    %Holds the k-nearest neighbors classification
    K = zeros(6,6);

    %Loop through each distance to every other song
    for i=1:length(d)
        [sortvals, I] = sort(d(:,i), 'descend');

        %Extract the k nearest neighbors (not including itself)
        nn = I(2:k+1);

        %Convert those to a genre index
        gi = floor(1+((nn-1)/25));

        %Loop through each genre for matches
        for j=1:6
            junk = size(gi(gi==j));
            num_in_genre(j) = junk(1);
        end


        %Find the majority matched genre
        [maxval, genre_index] = max(num_in_genre);

        %Increment the majority winner for the classification matrix
        if mod(i,25)==0
            K(floor(i/25),genre_index) = K(floor(i/25),genre_index) + 1;
        else
            K(floor(i/25)+1,genre_index) = K(floor(i/25)+1,genre_index) + 1;
        end
    end
end
```

Assignment 9

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Cross Validation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function CV = crossValidation(d)
    % Declare the test set and training set
    test_set = zeros(1,30);
    training_set = zeros(1,120);
    d2s = zeros(1,120);
    CV = zeros(6,6,50);

    %Perform 10 experiments
    for e=1:10
        %Randomize all the tracks
        rand_set = zeros(6,25);
        for i=1:6
            rand_set(i,:) = randperm(25);
            rand_set(i,:) = rand_set(i,:) + (i-1)*25;
        end

        %Select each generated test set
        for k=1:5
            %Generate row vector of test and training set
            test_set = reshape(rand_set(1:6,((k-1)*5 + 1:(k-1)*5+5)),[1,30]);
            training_set = setdiff((1:150),test_set);
```

```

            %Compare the k-nearest neighbors of each track in test set to each
            %in training set
            for test_index=1:30
                %Find the distance from test song at test_index to song l in
                %training set
               for l=1:120
                   d2s(l) = d(test_set(test_index), training_set(l));
               end

               %Sort the results
               [sortvals, I] = sort(d2s, 'descend');
               index_vals = training_set(I);

               %Extract the k nearest neighbors (not including itself)
               nn = index_vals(1:5);
               %Convert those to a genre index
               gi = floor(1+((nn-1)/25));

               %Loop through each genre for matches
               for j=1:6
                   junk = size(gi(gi==j));
                   num_in_genre(j) = junk(2);
               end

               %Find the majority matched genre
               [maxval, genre_index] = max(num_in_genre);

               %Increment the majority winner for the confusion matrix
               if mod(test_set(test_index),25)==0
                    CV(floor(test_set(test_index)/25),genre_index,(e-1)*5 + k) = ...
                        CV(floor(test_set(test_index)/25),genre_index,(e-1)*5 + k) + 1;
               else
                    CV(floor(test_set(test_index)/25)+1,genre_index,(e-1)*5 + k) = ...
                        CV(floor(test_set(test_index)/25)+1,genre_index,(e-1)*5 + k) + 1;
               end
            end
        end
    end
end
```

Assignment 11

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 4532: DSP Lab
% Lab 3: Improved Classification
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function CV = improvedClassification(d, mu, cov)
    % Declare the test set and training set
    test_set = zeros(1,30);
    training_set = zeros(1,120);
    d2s = zeros(1,120);
    CV = zeros(6,6,5);

    training_set_values = zeros(120,156);
    test_set_values = zeros(1,156);

    %Perform 1 experiments
    for e=1:1
        %Randomize all the tracks
        rand_set = zeros(6,25);
        for i=1:6
            rand_set(i,:) = randperm(25);
            rand_set(i,:) = rand_set(i,:) + (i-1)*25;
        end

        %Select each generated test set
        for k=1:5
%             %Generate row vector of test and training set
            test_set = reshape(rand_set(1:6,((k-1)*5 + 1:(k-1)*5+5)),[1,30]);
            training_set = setdiff((1:150),test_set);

            %Generate training matrix
            for x=1:120
               training_set_values(x,1:12) = mu(:,training_set(x))';
               training_set_values(x,13:156) = reshape(cov(:,:,training_set(x)),[1,144]);
            end

            %Create the Model
            Mdl = fitcecoc(training_set_values,training_set');

            %Test the model against all tracks in the test set
            for y=1:30
                test_set_values(1,1:12) = mu(:,test_set(y))';
                test_set_values(1,13:156) = reshape(cov(:,:,test_set(y)),[1,144]);
                predictResult = predict(Mdl,test_set_values);
                genre_index = floor(predictResult/25)+1;
```

```
                %Avoid Overflow
                if genre_index == 7
                    genre_index = 6;
                end
                %Increment the confusion matrix with the determined genre
                %index from the predict() function
               if mod(test_set(y),25)==0
                    CV(floor(test_set(y)/25),genre_index,(e-1)*5 + k) = ...
                        CV(floor(test_set(y)/25),genre_index,(e-1)*5 + k) + 1;
               else
                    CV(floor(test_set(y)/25)+1,genre_index,(e-1)*5 + k) = ...
                        CV(floor(test_set(y)/25)+1,genre_index,(e-1)*5 + k) + 1;
               end
            end

        end
    end
end
```