AMATH 482 Winter 2020
Homework 4: Music Classification

Ozan Erdal

March 6, 2020

Introduction and Overview

Machine Learning (ML) is an incredibly active field in the realm of mathematics and statistics. There exist many applications of ML to complete tasks that could be accomplished by a human, often times for later use with inhumanely sized data sets.
For the purpose of this experiment, ML and *Linear Discriminant Analysis* (LDA) were used in conjunction to accomplish the task of training a model that can classify music by its genre or artist. This is not a generally difficult task for a human to accomplish given an individual song and an active knowledge of music. In fact, if given a few songs to listen and asked to classify a brand new song into one of the categories, most people would succeed as the human auditory system is incredibly powerful.
Although it may not seem as such, this task is non-trivial for a computer to accomplish, but if achievable can be used to classify massive amounts of music at once, a task that might take years to complete for a human.

Theoretical Background

This experiment was split into 3 parts with very similar tasks but with different data sets and classification specificity:

- Test 1: Given three artists of different genres, classify a song from one of the three artists.
- Test 2: Given three artists of the same genre, classify a song from one of the three artists.
- Test 3: Given many artists from 3 different genres, classify a song from one of the three genres.

The Data

The underlying data was a large list of short sound files from the songs. However, since the techniques later quickly result in massive matrices often too large for even MATLAB to work with, the audio must be "sampled" to address this issue. Sampling in this context simply refers to taking smaller pieces of the audio files in hopes of accurately representing the song as a whole. For the purposes of this experiment, 4 different 5-second samples were picked from different parts of each song. Additionally, these tracks were sampled at a lower rate than the original audio.
The sampling rate standard for audio is $44100Hz$ as a direct result of the limits of the human auditory system. The upper range of hearing is just under half of that number, and according to the*Nyquist-Shannon sampling theorem*, this ratio is the minimum such that information is not lost. Audio signals are just a single example of this theorems reach, but it deals heavily with Fourier transforms and is thus relevant to all of signal processing. However, this theorem simply explains the reasoning behind a very common sampling frequency being $44100Hz$.
This is a simple stumbling block as the waveform does not need to be sampled at every point to acquire meaningful results from the subsequent experimentation. For the purposes of this experiment, sampling every 4 points was sufficient to significantly improve performance.
The data must then be separated into two sets for ML, a training and a testing set. Songs were alternatingly designated to each set, odd numbered tracks were in the training set and even numbered tracks in the testing set. The training set is used to determine thresholds with which the testing set can be classified into different groups by.

For Test 1, music from Aerosmith (Rock), Miles Davis (Jazz), and Louis the Child (Electronic) were used.
For Test 2, music from Pink Floyd (Classic Rock), Black Sabbath (Classic Rock), and Deep Purpled (Classic Rock) were used.
For Test 3, a large variety of music was used, but the categories were Rock, Disco, and Jazz.

The Technique

A combination of time-frequency analysis and LDA were employed to determine the thresholds for music classification.
Spectrograms of each song were constructed and reshaped into a column vector. For each grouping, all of these column vectors were then placed side-by-side resulting in an $m \times n$ matrix of size *number of samples* $\times$ *number of songs*. Since there are three classification groups, there are three of these matrices on which LDA is performed.
LDA relies on determining the best possible projection for the principal component projections. By maximizing the spread between points of the sample class and maximizing the distance between different classes' means, LDA splits up the data into groups and provides a projection that results in this split for further use with other data. This is the key as this projection is stored and subsequently

used on the training set.
These between and within equations are as follows:

$$S_B = \sum_{j=1}^{3} +m_j * (\overrightarrow{\mu_j} - \overrightarrow{\mu}) * (\overrightarrow{\mu_j} - \overrightarrow{\mu})'; \tag{1}$$

$$S_W = \sum_{j=1}^{3} +m_j * (\overrightarrow{x} - \overrightarrow{\mu}) * (\overrightarrow{x} - \overrightarrow{\mu})'; \tag{2}$$

For two classes the projection is to a single dimension, but with three classes, there are two projection vectors. When the data is easily separable, the first projection is sufficient, but since the projection matrix is required for later tests and the code is very similar, both projections were used in all tests. Although this process is quite straightforward, there is a constant that requires refinement to find the best value. The number of "features" determines how much of the SVD is used. This is similar to how different rank approximations can be used. 15 features was determined through trial and error to provide a good level of separation.
In a two-class classification, a threshold value can be set as the midpoint of the means of the two classes, but for three-class classification, lines of equidistant mean can be used as a separation technique. There are many ways to set threshold values but since other techniques were not used, they are unnecessary to discuss at length. The point of the threshold is a value for which to then compare the testing set projection to and classify the testing data as above or below each threshold. After classifying each song, the accuracy of the classification can be determined by looking at the number of discrepancies between the actual correct classifications which are known. Since the correct "answers" to the classifications are known, this is known as *supervised* ML and is essential for LDA. A plot of a single projection and full projection LDA's are shown below for a very small sample of music from classical, house, and rock music (none of which were used in the actual experiment).

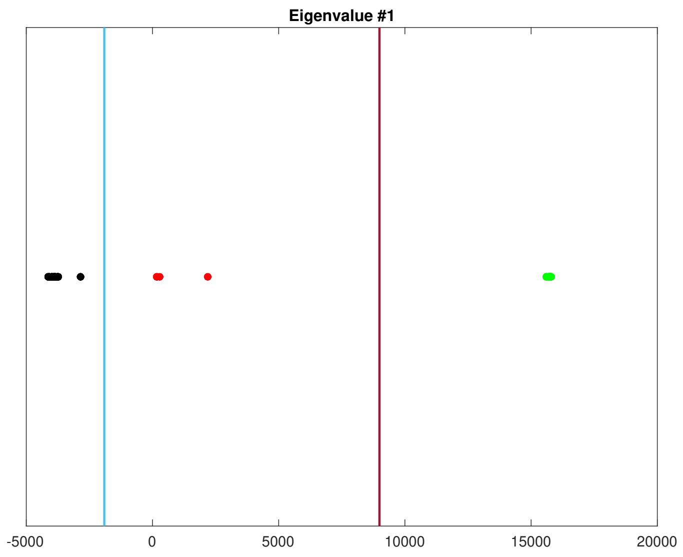

Figure 1: 1-dimensional LDA.

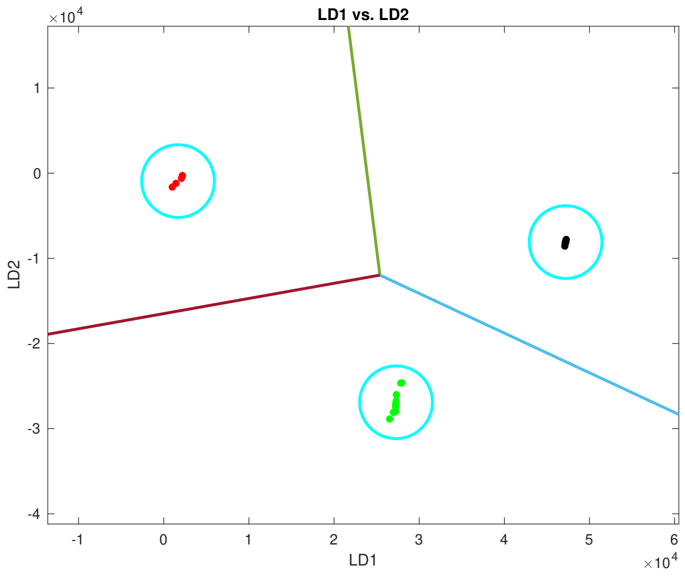

Figure 2: 2-dimensional LDA.

Algorithm Implementation and Development

For each song, the files were read in and converted to a waveform. Since the audio files were all stereo format, they had two channels, so the two channels were averaged to get a column vector. A function was written to accomplish this task since this process was repeated over and over. The result of this function was the sample file with the sampling process done as highlighted in the previous section. The samples were taken at the *20*, *40*, and *60* second marks in the songs. On some occasions, there would be songs that were too short for this process, so they were deleted and removed from any further consideration as they were likely an interlude or speech, unrepresentative of the actual classification.
These were systematically Fourier transformed with a filter width of 10 since this was a good value from past work with audio. The spectrograms had LDA performed, and the means of each cluster were caclulated and used to calculate points of equidistance, namely the center point of the entire LDA. Threshold lines were drawn from the center through each of the points of equal mean between respective clusters and beyond outside of the bounds of the plot area.
Based on these thresholds, the points were categorized as being from a particular region by looking at which threshold lines the testing set data lies above or below.

Computational Results

For each test, the 2-D LDA's are shown below and the points that were incorrectly classified were circled in cyan.
Training data is shown as dots and testing data is shown as plus signs.

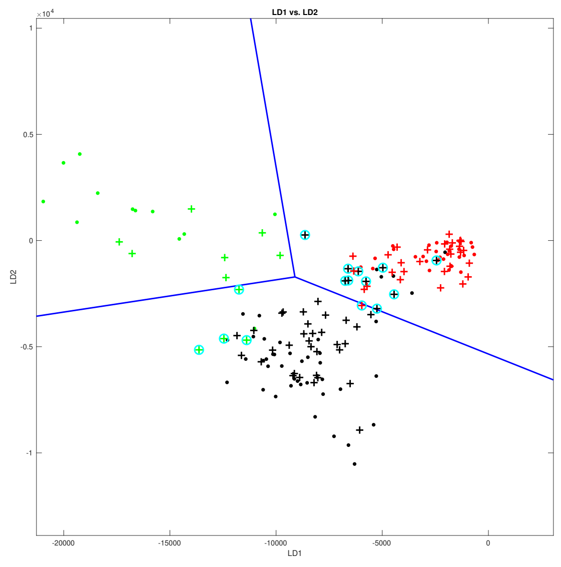

Figure 3: Test 1 LDA.

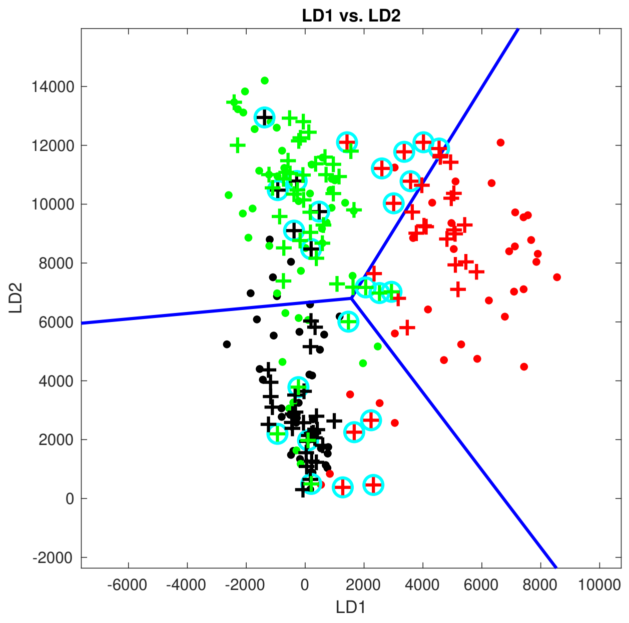

Figure 4: Test 2 LDA.

Summary and Conclusions

From this experiment, it was concluded that although only frequency information was provided to the SVD and the LDA, by performing these procedures in a particular way, nontrivial information was ascertained with a fairly good accuracy considering the simplicity of implementation and level of math behind the techniques.
All tests had well over 50% accuracy for classification correctness.

Appendix A

eig(S_1,S_2)

Calculates the generalized eigenvalue problem for the two matrices.
Outputs the eigenvalue and eigenvector matrices.

Appendix B

Sampling Music

```
function sample = adj_music(path,sec_len,start_pt,sps,rate,num_samples)
    sample = zeros(rate*sec_len,sps);
    [y,Fs] = audioread(path);
    for s = 1:sps
        if (size(y,2) > 1)
            y = mean(y,2); % convert to mono
        end
        sec_int = linspace(Fs*start_pt*s,Fs*(start_pt+sec_len)*s,num_samples+1);
        sec_int = sec_int(1:num_samples);
        sec = y(sec_int); % crop sample
        sample(:,s) = sec;
    end
end
```

Spectrogram Generation

```
function bandData = band_spec(bands,rate)
    [m,n] = size(bands); % samples x songs

    for i = 1:n
        % Construct Spectrogram
        v = bands(:,i)';
        L = m/rate;
        ts = linspace(0,L,m+1); t = ts(1:m);
        a = 10;
        tslide = linspace(0,L);
        vgt_spec = zeros(length(tslide),m);

        % Compute Spectrogram
        for j = 1:length(tslide)
            g = exp(-a*(t-tslide(j)).^2);
            vg = v.*g;
            vgt = fft(vg);
            vgt_spec(j,:) = abs(vgt);
        end

        bandData(:,i) = reshape(vgt_spec',m*length(tslide),1); % each column is a
    spectrogram
    end
end
```

SVD and LDA

```
function [vb1,vb2,vb3,W,U,S,V,mv,eq,center] = band_trainer(b1_spec,b2_spec,
   b3_spec,feature)
    nb1 = size(b1_spec,2);
    nb2 = size(b2_spec,2);
    nb3 = size(b3_spec,2);
    nbs = [nb1,nb2,nb3];

    [U,S,V] = svd([b1_spec,b2_spec,b3_spec],'econ');

    bs = S*V';
    U = U(:,1:feature);
    b1 = bs(1:feature,1:nb1);
    b2 = bs(1:feature,nb1+1:nb1+nb2);
    b3 = bs(1:feature,nb1+nb2+1:nb1+nb2+nb3);

    mbs = mean(bs(1:feature,:),2);
    mb1 = mean(b1,2);
    mb2 = mean(b2,2);
    mb3 = mean(b3,2);
    means = [mb1,mb2,mb3];

    % Scatter Within
    Sw = 0;
    for x = 1:nb1
        Sw = Sw + (b1(:,x)-mb1)*(b1(:,x)-mb1)';
    end

    for x = 1:nb2
        Sw = Sw + (b2(:,x)-mb2)*(b2(:,x)-mb2)';
    end

    for x = 1:nb3
        Sw = Sw + (b3(:,x)-mb3)*(b3(:,x)-mb3)';
    end

    % Scatter Between
    Sb = 0;
    for j = 1:3
        Sb = Sb + nbs(j)*(means(:,j) - mbs)*(means(:,j) - mbs)';
    end

    [V2,D] = eig(Sb,Sw);
    [~,ind] = maxk(abs(diag(D)),2);
    W = V2(:,ind); W = W/norm(W,2);

    vb1 = W'*b1; vb2 = W'*b2; vb3 = W'*b3;

    % reorder proj by param order
    mv1 = [mean(vb1(1,:),2),mean(vb2(1,:),2),mean(vb3(1,:),2)];
    mv2 = [mean(vb1(2,:),2),mean(vb2(2,:),2),mean(vb3(2,:),2)];

    mv = [mv1;mv2];

    center = [mean(mv1);mean(mv2)];
    m12 = [mean(mv1([1,2]));mean(mv2([1,2]))];
    m13 = [mean(mv1([1,3]));mean(mv2([1,3]))];
    m23 = [mean(mv1([2,3]));mean(mv2([2,3]))];
    v1 = m12 - center;
    v2 = m13 - center;
    v3 = m23 - center;
    u = [v1/norm(v1),v2/norm(v2),v3/norm(v3)];
    d = linspace(1,0.5e+05,2);
    eq = [center+u(:,1)*d;center+u(:,2)*d;center+u(:,3)*d];
```

```
end
```

Classification

```
for i = 1:size(testmat,2)
    song = pval(:,i);
    g = threshold(eq,center,song);
    resvec(i) = g;
end
diff = [resvec-labels];
err = [];
for d = 1:length(diff)
    if (abs(diff(d)) > 0)
        err = [err,d];
    end
end
```