AMATH482 HW05

Joshua Borgman

University of Washington

Josh Borgman

Friday, March 13, 2015

Contents

1. Introduction and Overview

Linear Discriminant Analysis is most frequently utilized as a dimensionality reduction tool to pre-process data meant for pattern/group classification purposes. In general terms, LDA seeks to project a dataset onto a low dimensional space with good separation of classes and a reduction of computational costs. LDA is categorized as a supervised algorithm, which learns from user input, and represents the axes that maximize the separation between multiple classes. Figure 1 represents a caricature of the results previously mentioned for LDA.

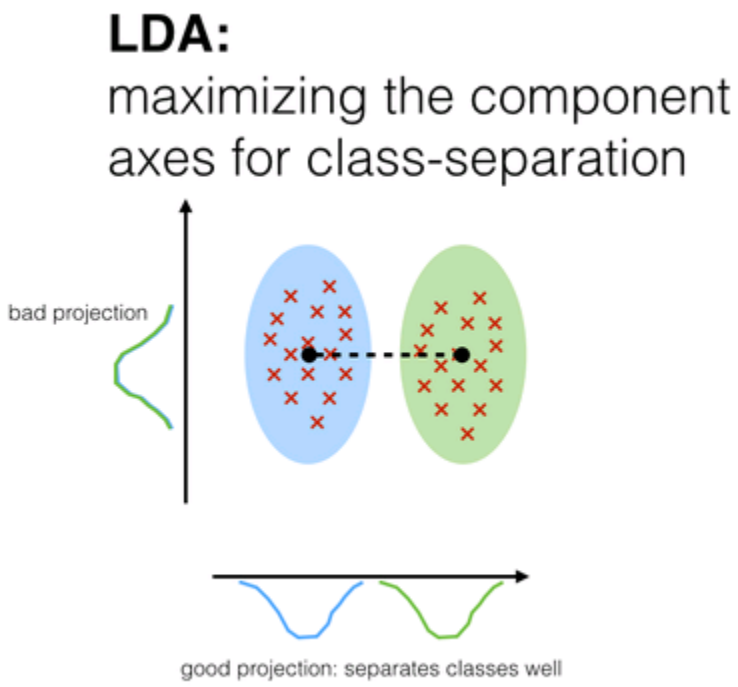

Figure 1: Diagram of LDA. Larger class mean separations and lower interclass variability is desired. Image is courtesy of Sebastian Raschka. (http://sebastianraschka.com/Articles/2014_python_lda.html)

The use of LDA on larger data sets requires a bit of finesse to permit the computationally expensive eigenvalue decomposition. Thus we will see the use of utilizing alternative representations of data through methods previously covered and reviewed below in the next section. These include singular value decomposition, principal components (analysis), and eigen equations.

2. Theoretical Background

Basics of Singular Value Decomposition

The reduced single value decomposition represents a transformation $\boldsymbol{A}$ between two orthonormal basis $\boldsymbol{v}_j$ and $\boldsymbol{u}_j$, such that

$$\boldsymbol{A}\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j \quad 1 \le j \le n.$$

Or in compact matrix notation $\boldsymbol{AV} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}$ or rearranging this to yield $\boldsymbol{A} = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\boldsymbol{V}^*$**.**

Traditionally the reduced single value decomposition is augmented to construct a matrix $\boldsymbol{U}$ from $\widehat{\boldsymbol{U}}$, by adding and additional *m-n* columns that are orthonormal to the already existing set in $\widehat{\boldsymbol{U}}$, so matrix **U** then becomes an *m x m* unitary matrix. This is followed by adding an additional *m – n* silent rows. This leads to $\boldsymbol{\Sigma}$ having *r* positive diagonal entries, with the remaining *n – r* diagonals equal to zero. The full SVD decomposition then takes the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$$

And the following three matrices

$$\boldsymbol{U} \in \mathbb{C}^{m\times m}\ is\ unitary$$

$$\boldsymbol{V} \in \mathbb{C}^{n\times n}\ is\ unitary$$

$$\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}\ is\ diagonal$$

Useful theorems that concern the single value decomposition include the following

1. Every matrix $\boldsymbol{A} \in \mathbb{C}^{mxm}$ has a singular value decomposition for the form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$, for which the singular values (σ_j) are uniquely determined and if A is a square and the singular values distinct, the singular vectors $\{u_j\} and \{v_j\}$ are uniquely determined up to a complex sign.
2. *If the rank of **A** is r, then there are r nonzero singular values.*
3. *range(**A**) =* $\langle \boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_r \rangle$ *and null(**A**) =* $\langle \boldsymbol{v}_{r+1}, \boldsymbol{v}_{r+2}, \dots, \boldsymbol{v}_n \rangle$
4. $\|\boldsymbol{A}\|_2 = \sigma_1$ and $\|\boldsymbol{A}\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2}$ where these are the L2 and Frobenius norms respectively. The Frobenius norm contains the total matrix energy, while the 2-norm definition contains the energy of the largest singular value.
5. *The nonzero singular values of **A** are the square roots of the nonzero eigenvalues of **A*A** (or **AA***).*
6. *If **A=A*** (self adjoint), then the singular values of **A** are the absolute values of the eigenvalues of **A**.*
7. *For* $\boldsymbol{A} \in \mathbb{C}^{mxm}$*, the determinant is given by* $|\det(A)| = \prod_{j=1}^{m} \sigma_j$*. This result stems from the fact that the matrices **U** and **V** are unitary matrices.*

Based on these theorems, we can understand the SVD as a sort of least-square fitting algorithm, allowing us to project the matrix onto low dimensional representations in a formal way.

More formally, we can represent the sum of r rank-one matrices as

$$\mathrm{A} = \Sigma_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*,$$

And partial sum which considers N, such that $0 \le N \le r$, can be represented as

$$\mathrm{A_N} = \Sigma_{j=1}^{N} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*$$

Thusly, if $N = \min\{m, n\}$, we can define $\sigma_{N+1} = 0$, and

$$\|A - A_N\|_2 = \sigma_{N+1}$$

Finally it is worth emphasizing that the SVD produces characteristic features (principal components) that are determined by the covariance matrix of the data.

Eigenvalues, Eigenvectors and Diagonalization

The simpler alternative to SVD most related to linear algebra is the eigenvalue decomposition. In particular eigenvalues and eigenvectors have many applications in understanding differential equations. The equation

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{A}\boldsymbol{y}$$

can be assumed to have a solution of the form $y = \exp(\lambda t)$ which leads the eigen value problem:

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

Which we will solve by reforming the equation as follows.

$$\boldsymbol{A}\boldsymbol{x} - \lambda \boldsymbol{I}\boldsymbol{x} = (\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{x} = 0$$

We can thus find the eigenvalues (λ_j) and the associated eiegenvectors $\boldsymbol{x}$ which satisfy the eigenvalue problem.

One major benefit of the eigenvalue decomposition is its ability to decompose a matrix (here **A**) into the following form

$$\boldsymbol{A} = \boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{S}^{-1}$$

Where $\boldsymbol{S}$ the matrix here is the columns are the eigenvectors of **A,**

$$\boldsymbol{S} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n})$$

And the matrix $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonals are the corresponding eigenvalues, namely:

$$\begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{pmatrix}$$

This diagonalization leads the useful property that $\boldsymbol{A}^M = \boldsymbol{S}\boldsymbol{\Lambda}^M\boldsymbol{S}^{-1}$

Principal Component Analysis (PCA)

Principal Component Analysis, in combination with techniques such as SVD are widely used in dimensionality reduction of data. PCA offers users a low-dimensional projection of the viewed from its most informative viewpoints. This is accomplished by utilizing the first few principal components of the data so that the dimensionality of the transformed data is greatly reduced.

Proper Orthogonal Modes /Principal Components

The proper orthogonal modes are the representative modes generated from a single value decomposition. These modes are based on the L2 fitting of the covariance matrix of the data set. This unique set of basis functions are all orthogonal and represent the principal components of the analyzed system.

Linear Discriminant Analysis

The main ideas of Linear Discriminant Analysis were initially suggested by Fisher in regards to taxonomy. As it can be extended successfully to statistical analysis and classifications, the concept is a useful form of supervised learning and classification. Two or more data sets are considered and projected onto a new basis in LDA. The goal of LDA is to optimize this projection such that we can maximize the distance between the inter-class data while minimizing the intra-class data. This abstraction leads to the idealized projection $\boldsymbol{w}$, defined as

$$\boldsymbol{w} = \arg\max\left(\frac{\left(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}\right)}{\left(\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}\right)}\right.$$

Where the intra-class scatter matrix $\boldsymbol{S}_B$ and the inter-class scatter matrix $\boldsymbol{S}_W$ is given by

$$\boldsymbol{S}_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T$$

$$\boldsymbol{S}_W = \Sigma_j^2 \Sigma_{\boldsymbol{x}} \left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)\left(\boldsymbol{x} - \boldsymbol{\mu}_j\right)^{\boldsymbol{T}}$$

x is the group data set and **j** is the number of groups that you are considering. For reference, the formulation of **w** above is known as the generalized Rayleigh quotient. The solution of this equation can be found by the generalized eigenvalue problem

$$\boldsymbol{S}_B \boldsymbol{w} = \boldsymbol{\lambda} \boldsymbol{S}_W \boldsymbol{w}$$

For which the maximum eigenvalue λ and its associated eigenvctor gieve the quantity of interest and the projection basis.

3. Algorithm Implementation and Development

Generalized Procedure

1. Sample various clips of music – create matrix of song sample clips
2. Perform Gabor (STFT) Transform on each clip – Yields spectrogram data
3. Form Data Matrix by reshaping and collecting spectrogram data.
4. Perform SVD on data matrix – decompose into modes that are easier to analyze.
5. Compute Scatter Matrices
6. Perform Eigenvalue decomposition
7. Construct “w” projection.
8. Compare groupings – Optional computation of threshold value between groups.

Principal Component Decomposition

The principal component analysis was performed using the following generic steps:

1. Organize the data into a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ where *m* is the number of measurement types and *n* is the number of measurements (or trials) taken from each probe.
2. Subtract off the mean for each measurement type or row $\boldsymbol{x}_j$.
3. Compute the SVD and singular values of the covariance matrix to determine the principal components.

Single Value Decomposition

The Single value decomposition is accomplished through the use of the svd command which decomposes the vector into the matrices previously discussed. Namely:

$$[U, S, V] = svd(A);$$

yields the single value decomposition for A, with basis vectors given by U and V, and diagonal singular value in S. To assemble the matrix A for our operation we used the form in which the columns are data from a given spectrogram

$$X = (\boldsymbol{Song1Clip1}, \boldsymbol{Song1Clip2}, \dots \boldsymbol{Song3Clip1}, \dots, \boldsymbol{SongNClipM})$$

Using this form we are able to compute the SVD to find a lower dimensional representation that can be used in the eigenvalue problem.

Computing Scatter Matrices

$\boldsymbol{S_W}$ was computed using the formulation stated above. Each clip was subtracted by its respective class mean and then squared by multiplying by the transposed matrix. The results were then added to the other groups, previously treated in the same manner.

$\boldsymbol{S_B}$ was computed by subtracting each possible pair of means ({1,2},{1,3},{2,3}), such that the intra-class variability was accounted for. This subtraction was squared by multiplying by the results transpose, as detailed above.

Construction of "w" Projection

The construction of $\boldsymbol{w}$ continued by solving the previously stated eigenvalue equation, yielding a set of eigenvalues and vectors. Selecting the largest eigenvalue we locate its corresponding eigenvector, which is the un-normalized $\boldsymbol{w}$**.** **W**e then normalize by dividing by the norm of $\boldsymbol{w}$.

Comparing Groups

Histograms of each clip's projection by **w** show visual representations of the effects typical of LDA. Thresholds between groups were computed by averaging separations.

4. Computational Results

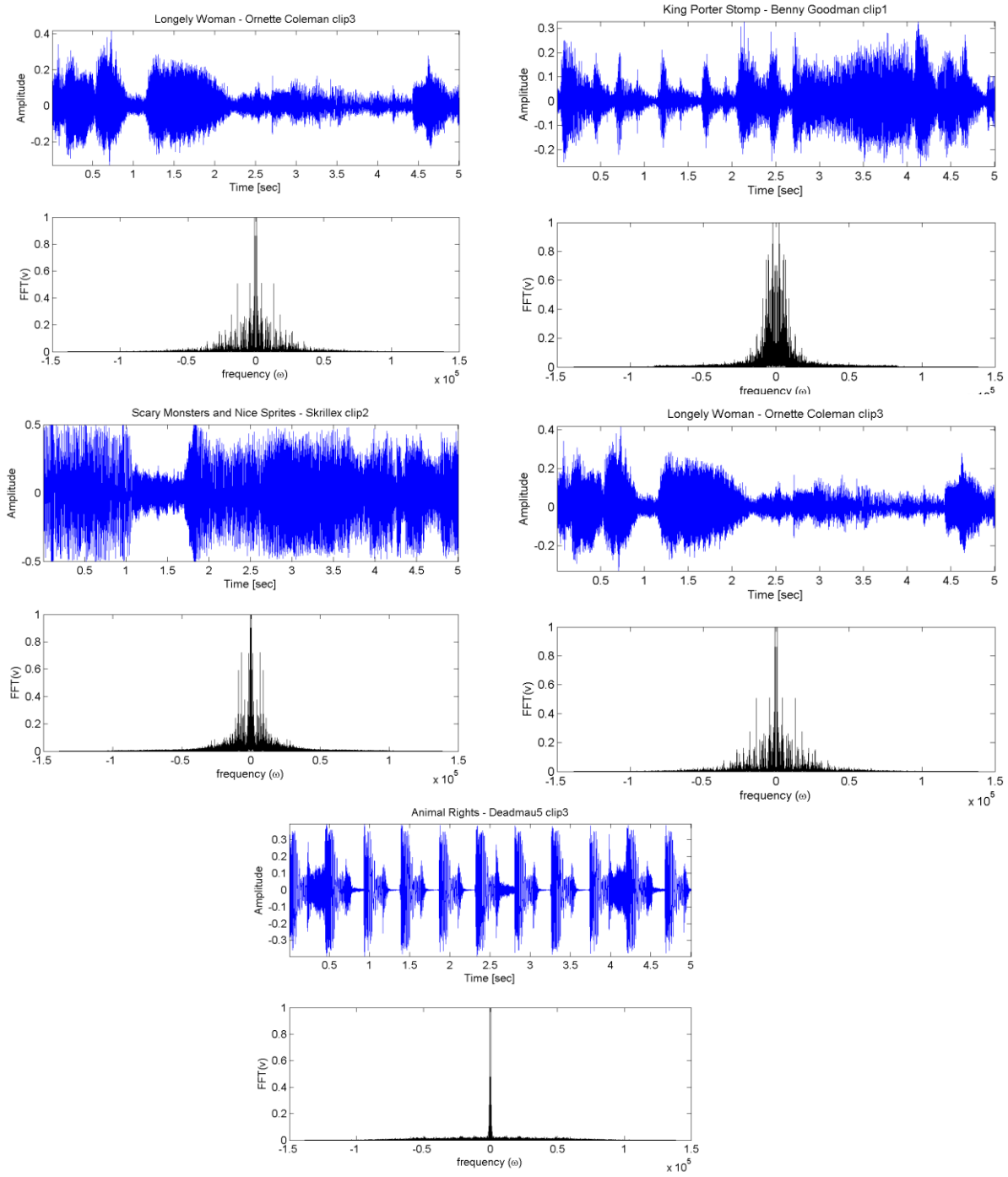

Figure 2: The plots above show a variety of styles of music by several artists. Note the distinct differences seen in sections of their music as well as the varied frequency contents.

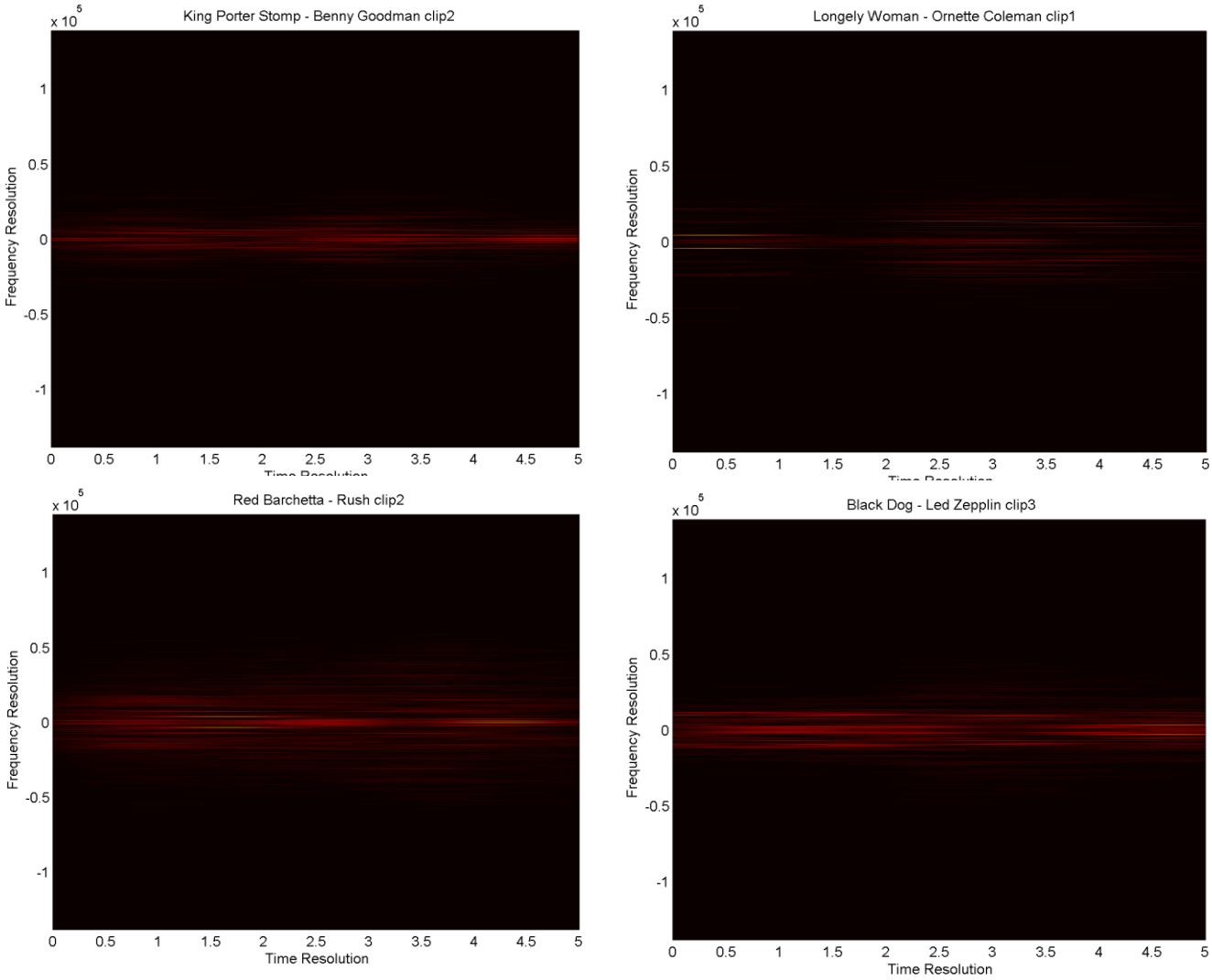

Figure 3: Visualized Spectrograms of some song clips. The spectrogram data is later used to define the groups.

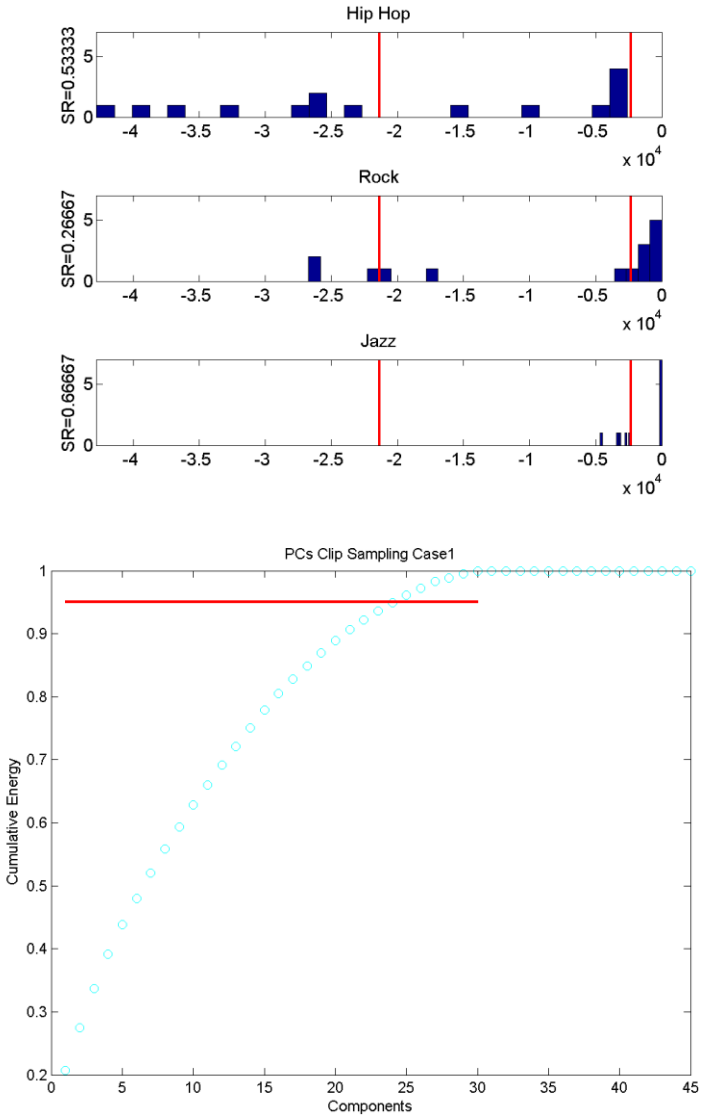

Figure 4: The above graphic is a visualization of the separation of projected clips and threshold lines that should separate groups. Below is a plot of cumulative energies of SVD modes. Note that here 45 clips, 15 from each genre artist (Macklemore, Led Zeppelin, Ornette Coleman).

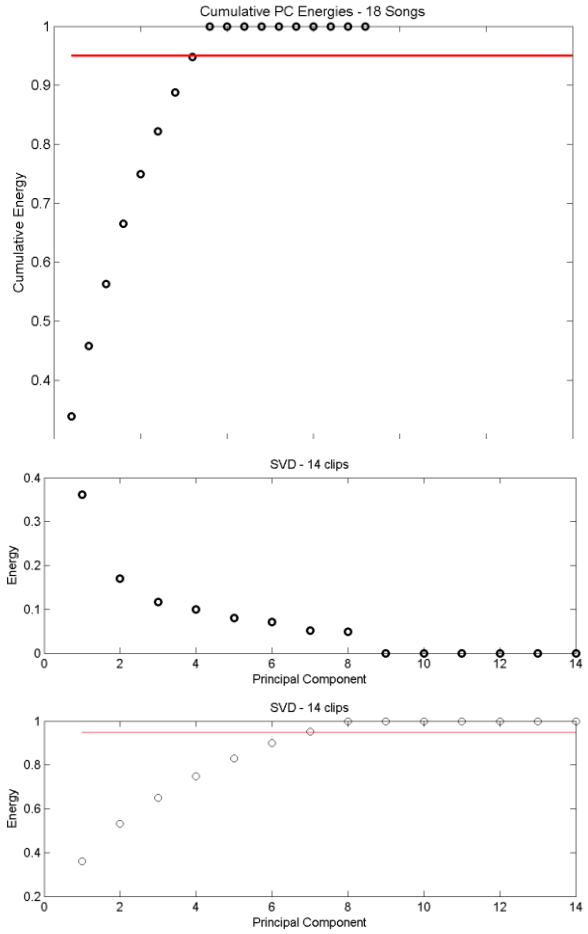

Figure 5: It is worth noting the effect that the number of songs had. The reduced information made for a very difficult separation of classes, yielding very poor success rates. Above is the cumulative energy content of the case 1 with fewer songs and clips. Both plots are based on Case 1

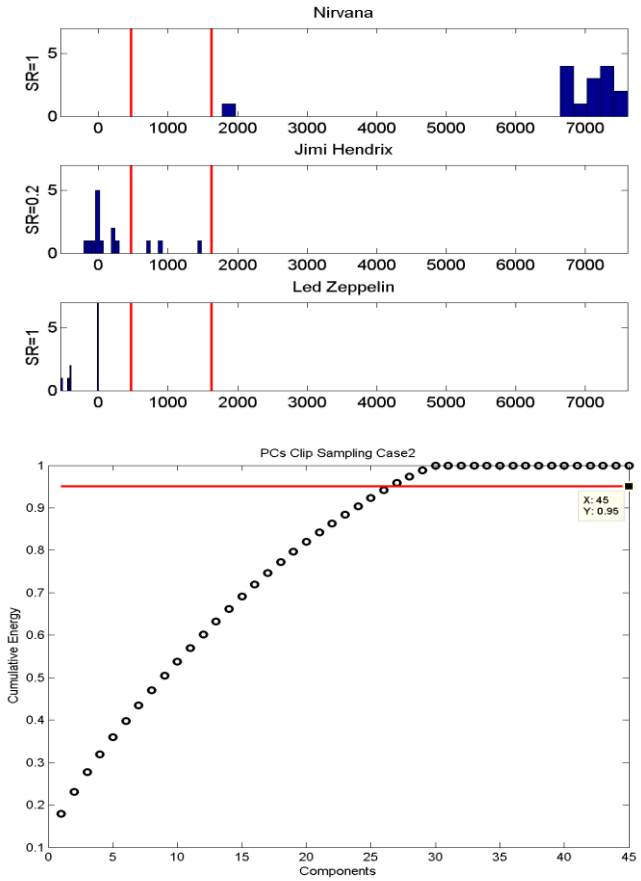

Figure 6: Above are the results from Case 2 - A strong separation between two groups but a muddling of two proved slightly more difficult than bands of different genres.

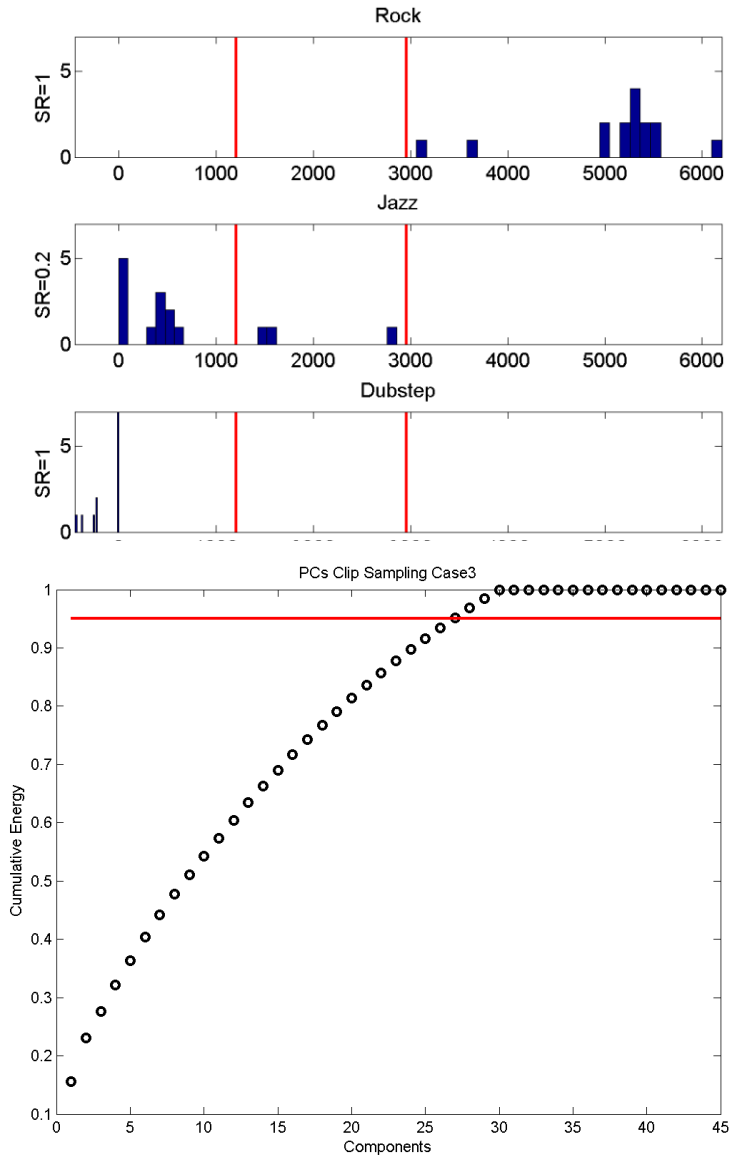

Figure 7: Above are the results for Case 3 - Again it appears that Jazz is hard to classify distinctly from groups. There is also a great deal of localization for Dubstep music. Reasonably so, since it all kinda sounds the same.

5. Summary and Conclusion

The use of supervised classification seen here in is a perfect utilization of the LDA algorithm to try and separate categories. Of course as is the case with any classification problem, we are limited in our ability to draw fine thresholds between genres, groups, or classes as our statistical learning sets are limited. By increasing our amount of data we are able to improve on our separations and classifications. This limitation on data comes not only in the availability of music and the subjective classifications we do pre-processing the groups, but also by our computing power. These algorithms, particularly the assembly of the data matrix from spectrograms, the SVD, and eigenvalue computations are computationally expensive. Here we have experimented with reducing redundant data (taking only half the frequencies in the FFT since they are symmetric), using the 'economic' version of the SVD, and resampling our data sets to reduce their size and computational expense. Reducing the number of songs to make our data more computationally friendly simply yields data that is hard to discriminate with any certainty.

In ending, linear discriminant analysis, is a useful procedure for dimensionality reduction and compression of data sets, as well as providing a basis that can be exploited for logical decision making and differentiation between pre-defined classes.

Appendix A:

Appendix B:

```
%{
Joshua Borgman  - AMATH 482 : Homework 5: Classification Algorithms.
03/05/2015

This section is used to find and extract data clips and then perform the
conversion to spectrograms for further analysis in dc_trainer.m
%}

clear all; close all; clc
%% Set Directory to Music Files

%   cd 'C:/Users/Josh Borg/Music/Rush/Chronicles Disc 2' %Rush%
% cd 'C:/Users/Josh Borg/Music/Led Zeppelin/Led Zeppelin IV' %Led Zeppelin%
% cd 'C:/Users/Josh Borg/Music/Jimi Hendrix Experience/Are you Experienced-
[US]' %Jimi Hendrix Experience%
%
%cd 'C:/Users/Josh Borg/Music/The Norton Jazz Recordings Disc 3' % King
Porter Stomp, Giant Steps, Longely Woman
%
%
% cd 'C:/Users/Josh Borg/Music/UKF Dubstep 2010 - MartN' % Flux Pavillion -
Cracks
% cd 'C:/Users/Josh Borg/Music/Deadmau5_-_4_x_4_Equals_12-CD-2010-SQ'
%Deadmau5 - Animal Rights
% cd 'C:/Users/Josh Borg/Music/Skrillex - Scary Monsters and Nice Sprites'
%Skrillex - Scary Monsters and Nice Sprites

% cd 'C:/Users/Josh Borg/Music/Macklemore/The Unplanned Mixtape'
%   The Town
%   And We Danced
%   Irish Celebration
% cd 'C:/Users/Josh Borg/Music/AC-DC/Back in Black'
%   Hells Bells
%   Shoot to Thrill
%   Let Me Put My Love Into You
% cd 'C:/Users/Josh Borg/Music/Yo-Yo Ma/Bach_ The Cello Suites'
%   Suite for Solo Cello No1 in Gm
%   Suite for Solo Cello No2 in Dm
%   Suite for Solo Cello No3 in Cm

% cd 'C:/Users/Josh Borg/Music/Nirvana/Nevermind'
%   Smells Like Teen Spirit
%   Come As You Are
%   Lithium
% cd 'C:/Users/Josh Borg/Music/Jimi Hendrix/Blues'
%   Red House
%   Voodoo Chile Blues
%   Born Under a Bad Sign
```

```
% cd 'C:/Users/Josh Borg/Music/Led Zeppelin/How the West Was Won Disc 1'
%   Immigrant Song
%   HeartBreaker
%   Since I've Been Loving You




%fs 44100
%% This generates 5 second clips

clips = [];

%% Create clips from current song

mono =  mean(data,2)';
clipLen = fs*5;
noClips = 5;
maxSam = (length(mono)-clipLen);
start = int32((maxSam -1).*rand(noClips,1)+1);

% [Clip1Song1] %[ N*M x 1]
% [...       ]
% [ClipNSongM]


for j = 1:noClips
    clip1 = mono(start(j):(start(j) + clipLen));
    clips = [clips; clip1];
end




%%
% Creates song matrix
song = cell(9,1);
%Rock n Roll
song{1} = 'Red Barchetta - Rush';
song{2} = 'Black Dog - Led Zepplin';
song{3} = 'Purple Haze - Jimi Hendrix Experience';

%Jazz
song{4} = 'King Porter Stomp - Benny Goodman';
song{5} = 'Giant Steps - John Coltrane';
song{6} = 'Longely Woman - Ornette Coleman';

%EDM - Dubstep
song{7} = 'Cracks - Flux Pavillion';
song{8} = 'Animal Rights - Deadmau5';
song{9} = 'Scary Monsters and Nice Sprites - Skrillex';
```

```
% Other Songs Used -
% Limelight - Rush
% Rock n Roll - Led Zeppelin
% Highway Chile - JHE
% Autumn Nocturne - Lou Donaldson
% El Matador - Kenny Dorham
% Boogie Stop Shuffle - Chrles mingus
% Piano Tune - Bar 9
% One Trick Pony (ft. Sofi) - Deadmau5
% Kill Everybody - Skrillex

%%
cd 'C:\Users\Josh Borg\Documents\AMATH 482\Homework 5\Song Waveforms and
Transforms'


%%    Creating Spectogram - This section will find FFTs and wave numbers
%

vAll =[];
kAll = [];
for j = 1:45
    v = clips(j,:)/2; %Necessary?
    vAll = [vAll; v];
    vt = fft(v);
    vt = vt(1:ceil((length(vt)/2)));

    n = length(v); L = 5;
    t =(1:n)/fs;
    k = (2*pi/L)*[0:n/2 -n/2:-1];
    k = k(1:ceil((length(k)/2)));
    ks = fftshift(k);
    kAll = [kAll; k];
%     fileName = strcat(song{floor((j-1)/3)+1}, ' clip ', num2str(mod(j-
1,3)+1));
%
%     figure(1)
%     subplot(2,1,1) %Time Domain
%     plot(t, v)
%     set(gca, 'Fontsize', [12])
%     xlabel('Time [sec]'), ylabel('Amplitude'), axis('tight');
%     title(fileName);
%
%     subplot(2,1,2)
%     plot(ks, abs(fftshift(vt))/max(abs(vt)),'k'); %nomralized by max
%     set(gca, 'Fontsize', [12])
%     xlabel('frequency (\omega)'), ylabel('FFT(v)')
%     print(fileName, '-dpng');
end

%%
cd 'C:\Users\Josh Borg\Documents\AMATH 482\Homework 5\Spectograms'
%%
cd 'C:\Users\Josh Borg\Documents\MATLAB'
%%
% %his section creates spectrogram and stores in a cell that corresponds to
```

```
% song and clip number
spect = cell(15,3);
for jj = 1:45
    songNum = floor((jj-1)/3)+1; clipNum = mod(jj-1,3)+1;
    %fileName = strcat(song{songNum}, ' clip ', num2str(clipNum));
    v = vAll(jj,:);
    vgt_spec=[];
    tslide=0:0.1:5;
    width = 1;
    for j=1:length(tslide)
        g=exp(-2*(t-tslide(j)).^2); % Gabor
        vg=g.*v; vgt=fft(vg); vgt=vgt(1:ceil((length(vgt)/2)));
        vgt_spec=[vgt_spec; abs(fftshift(vgt))];
    %     subplot(3,1,1), plot(t,v,'k',t,g,'r')
    %     subplot(3,1,2), plot(t,vg,'k')
    %     subplot(3,1,3), plot(ks,abs(fftshift(vgt))/max(abs(vgt)))
    %     axis('tight')
    %     drawnow
    %     pause(0.05)
    end

    spect{songNum,clipNum} = vgt_spec;

    %Generates Spectogram of Piece
%     figure(2)
%     pcolor(tslide,fftshift(kAll(jj,:)),vgt_spec.'), title(fileName),...
%         set(gca,'Fontsize', [12]), xlabel('Time Resolution'), ylabel(...
%         'Frequency Resolution');
%     shading interp
%     colormap(hot)
%     axis([0 5 -.75e5 .75e5])
%    print(fileName, '-dpng');
end



%{
Joshua Borgman  - AMATH 482 : Homework 5: Classification Algorithms. LDA
Procedure once you have spectrogram data
03/05/2015

%}
%%
%Set size parameters and create predefined data matrix, note option for
%'resampled' size
noClips = 3;
noSongs = 15;
nr = noSongs; nj = noSongs; nd = noSongs;

[m, n] = size(spect{1,1});
%data = zeros(ceil(0.7*m*n), noClips*noSongs);
data = zeros(m*n,noClips*noSongs);
%%
%Reshape spect data into data matrix, option to resample to reduce size
for j = 1:noSongs
```

```
    for jj = 1:noClips
%         data(:,j*jj) = resample(reshape(spect{j,jj},m*n, 1),7,10);
          data(:,j*jj) = reshape(spect{j,jj},m*n, 1);
    end
end
%%
% reduces number of songs and clips used
% data1 = data(:,[1,2,4,5,7,8,10,11,13,14,16,17,19,20,22,23,25,26]);
% data2 = data(:,1:2:end);
d2 = d1(:,1:2:end);
%% Another way to resample data if SVD computation fails

for j = 1:noClips*noSongs
    d1(:,j)=resample(data(:, j),8,10);
end
%%
% economic svd
[U, S, V] = svd(data, 0);
%% Plots PCs and 95% threshold line
figure(2), plot(cumsum(diag(S))/sum(diag(S)), 'ko', 'Linewidth', [2])
hold on, plot([1 45],[.95 .95], 'r-', 'Linewidth', [2]),
title('PCs Clip Sampling Case3'), xlabel('Components'), ylabel('Cumulative
Energy')
%%
%defines number of features to capture 95% of energy
feature = find(cumsum(diag(S))/sum(diag(S))>=0.95,1);
songs = S*V';
U = U(1:feature);

%%
%SVD projections and mean calculations
rock = songs(1:feature,1:nr);
jazz = songs(1:feature,nr+1:nr+nj);
dub = songs(1:feature,nr+nj+1:nr+nj+nd);


mr = mean(rock, 2);
mj = mean(jazz,2);
md = mean(dub,2);
%% Creates Scatter Matrices
Sw = 0; % within class variances
for j=1:nr
    Sw = Sw + (rock(:,j)-mr)*(rock(:,j)-mr)';
end
for j=1:nj
    Sw = Sw + (jazz(:,j)-mj)*(jazz(:,j)-mj)';
end
for j=1:nd
    Sw = Sw + (dub(:,j)-md)*(dub(:,j)-md)';
end

Sb = (mr-mj)*(mr-mj)' + (mr-md)*(mr-md)' + (mj-md)*(mj-md)';
%St = Sb + Sw;

%%
% linear discriminant analysis
```

```
[V2, D] = eig(Sb,Sw);


[lambda, ind] = max(abs(diag(D)));
w = V2(:, ind); w = w/norm(w,2);

vrock = w'*rock; vjazz=w'*jazz; vdub = w'*dub;

result = [vrock, vjazz, vdub];


sortRock = sort(vrock); sortJazz = sort(vjazz); sortDub = sort(vdub);
%%
%Define Thresholds
groups = [sortRock; sortJazz; sortDub];
gmeans  = mean(groups, 2);
totalmin = min(min(groups));
totalmax = max(max(groups));

group1 = groups(find(gmeans==max(gmeans)),:);
group3 = groups(find(gmeans==min(gmeans)),:);

group2 = groups(find(and(gmeans~=max(gmeans),gmeans~=min(gmeans))),:);
%%
t1 =1; t2 =length(group2);
while group3(t1)>group2(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold1 = (group3(t1)+group2(t2))/2;
t1 = 1; t2 =length(group2);
while group2(t1)>group1(t2)
    t1 = t1-1;
    t2 = t2+1;
end
threshold2 =  (group1(t1)+group2(t2))/2;
%% Succes Rates
srg1 = sum(group1>threshold2)/length(group1);
srg2 = sum(and(group2<threshold2, group2>threshold1))/length(group2);
srg3 = sum(group3<threshold1)/length(group3);
srg1, srg2,srg3
%% Make plot of separations, added threshold lines

figure(4)
subplot(3,1,1)
hist(sortRock,30); set(gca, 'Xlim', [totalmin totalmax], 'Ylim',
[0,7],'Fontsize', [14]), title('Nirvana'), hold on,
plot([threshold1 threshold1],[0 7],'r-', [threshold2 threshold2],[0 7], 'r-',
'Linewidth',[2]);

subplot(3,1,2);
hist(sortJazz,30);set(gca,'Xlim', [totalmin totalmax], 'Ylim', [0,7],
'Fontsize', [14]), title('Jimi Hendrix'), hold on,
plot([threshold1 threshold1],[0 7],'r-', [threshold2 threshold2],[0 7], 'r-',
'Linewidth',[2]);
```

```
subplot(3,1,3);
hist(sortDub,30);set(gca,'Xlim', [totalmin totalmax], 'Ylim', [0,7],
'Fontsize', [14]), title('Led Zeppelin'), hold on,
plot([threshold1 threshold1],[0 7],'r-', [threshold2 threshold2],[0 7], 'r-',
'Linewidth',[2]);
```