Time to Face the Music: Simple Machine Learning on Face and Music Data

Kristen Drummey

Department of Physiology and Biophysics, University of Washington

Github: github.com/kldrummey/AMATH582

Abstract

Machine learning has quickly become a popular method in automating analysis of large datasets. In this study, simple machine learning was performed on images of faces from the Yale Face Database and short samples from songs, to determine whether classifiers could be constructed that could assign novel data to the correct classification. Although classifiers were not able to be constructed for either data set in this study, basic qualities of the dataset were pulled out using singular value decomposition and linear discrimination analysis.

1 Introduction and Overview

Machine learning has exploded in popularity across scientific fields recently, with more and more researchers using machine learning algorithms to automate and classify a wide range of datasets, such as large volumes of neural activity data or sub-second recordings of animal behavior. Machine learning, whether supervised or unsupervised, is built upon decomposition, such as through wavelets, singular value decomposition (SVD), and linear discrimination analysis (LDA).Decomposing input data into its component waves allows for application of the SVD, which pulls out the principal components that represent the largest amounts of variance in the data. Finally, after applying the SVD to the data, the LDA can be applied to determine a statistical threshold for calling input data as classified in one category vs. another.

The first part of this study focuses on decomposing and classifying images of human faces that were compiled by the Yale Faces Database. The second part of this study attempts to build classifiers that distinguish novel clips of songs between bands and across genres.

2 Theoretical Background

LDA is an essential component that builds upon the wavelet and SVD methods previously discussed, and allows for statistical interpretations that are essential for building effective machine learning algorithms. After decomposition of input data by wavelet decomposition and SVD, LDA is applied in order to maximize the distance between data that is in different classes, allowing for separation between, for example, two songs by different artists, while also minimizing the distance between data in the same class - for example, grouping different songs by the same artist together.

LDA accomplishes this by constructing a projection, w (Eq. 1) that is based on scatter matrices for between class, S_b (Eq. 2) and within class, S_w (Eq.3). These matrices measure both the variance within the class, as well as the variance between class means.

$$w = argmax \frac{w^T S_B w}{w^T S_W w} \tag{1}$$

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{2}$$

$$S_W = \Sigma_{j=1}^{2} \Sigma (x - \mu_j)(x - \mu_j)^T \tag{3}$$

Together, this process provides a method of statistically looking at the properties of the input data, and the threshold for determining what data counts as being in one class versus another.

3 Algorithm Development and Implementation

For the first part of this study, images of human faces in different lighting and with different facial expressions were used. Data was loaded in and reshaped before undergoing SVD analysis.

Algorithm 1: Load facial data and perform SVD

Load 25 facial images for each dataset using `uiimport` and `imread`
Reshape facial data so that all facial data is a matrix, where each column indicates and image and each row contains pixel information for that image.
Find and subtract the mean and use `[u s v]=svd(images)` to perform SVD on cropped and uncropped datasets.

For the second part of this study, 5-10 second song clips from different artists and across genres were loaded in. A list of the songs, artists, and genres can be found in Table 1 below. After loading, a spectrogram was computed for all song clips.

Algorithm 2: Load in audio data and compute spectrograms

Load .mp3 song clips using `uiimport` and `audioread`
for $j = 1 : length(songclip_t slide)$ **do**
  Apply Gabor transform with width 25 and timestep 0.1 Multiply song clip data by Gabor transform
  Compute Fourier transform of the filtered song clip Add transformed data into spectrogram matrix
end for

After computing the spectrograms, song data was combined into one matrix and underwent SVD and LDA analysis.

Algorithm 3: SVD/LDA of song clips and determination of classifier accuracy

Find and subtract the mean and apply SVD to the collected song data using `[u s v]=svd(songclips)`
Apply LDA. Compute the projection, w, and the scatter matrices S_B and S_W
Compute the error and success rates of the algorithm in assigning novel song clips to the correct category.

4 Computational Results

Figure 1 shows the results of SVD analysis on the images from the Yale Faces Database. Panel A shows the energy captured by each mode for both the cropped (top) and uncropped faces, determined by the diagonal of the Σ matrix, while Panel B shows the projections of 10 images onto the first three modes. The first two modes of the cropped data account for a combined 75% of the variance, whereas the first two modes of the uncropped data only account for about 65% of the variance, suggesting that the cropped images need fewer modes to produce a more accurate facial reconstruction. Additionally, the projections onto the modes, using the V matrix, appear to show that different features are being picked up in the cropped vs. uncropped datasets. However, only 25 images from each dataset was used, so these conclusions should be interpreted with caution.

Figure 2A shows the percent of energy captured in each mode for the comparison between the Talking Heads, Mac DeMarco, and Stevie Wonder, all vastly different artists that encompass different genres. Figure

Genre	Artist	Song
New Wave	Talking Heads	Once in a Lifetime This Must Be The Place Psycho Killer
Alternative	Mac DeMarco	Dreamin' Freakin' Out the Neighborhood My Kind of Woman
Soul	Stevie Wonder	For Once In My Life Signed, Sealed, Delivered Superstition
	Marvin Gaye	Let's Get It On Sexual Healing What's Goin' On
	Earth, Wind, and Fire	September Shining Star Sing a Song
Hip Hop	Kendrick Lamar	Humble King Kunta Wesley's Theory
	Tyler, the Creator	EARFQUAKE I THINK NEW MAGIC WAND
	Vince Staples	Bag Bak Outside! Surf
"Sad Dad" Alternative	Bon Iver	33, GOD 666 Hey, Ma
	Local Natives	Bowery When Am I Gonna Lose You Wide Eyes
	The National	Bloodbuzz Ohio Conversation 16 Squalor Victoria

Table 1: Training set of songs used. Originally, the goal was to compare 1.) Bands across genres (Talking Heads vs Mac DeMarco vs Stevie Wonder), 2.) Bands within genre (Kendrick Lamar vs. Tyler, the Creator vs. Vince Staples), and 3.) Genres (Soul vs. Hip Hop vs. "Sad Dad" Alternative)

2B shows a histogram of the overlapping samples with the thresholds between samples shown as dotted lines on the graph.

Figure 3 shows a similar analysis, but for three artists within the same genre of hip hop (Kendrick Lamar vs Tyler, the Creator vs Vince Staples). The amount of energy captured in each mode is substantially less than that captured by the bands-between-genres analysis, and the thresholds between hip hop artists shown in Figure 3B are closer together than those seen in Figure 2B, suggesting that it is more difficult to capture differences between artists in the same genre.

Ideally, this study would have also included analysis of classification between genres, as put forward by the songs listed in Table 1. However, the author is horrendously bad at time management and was unable to perform genre classification or form test datasets. An attempt was made to classify a novel Mac DeMarco song ('Ode to Viceroy') and determine a confusion matrix of errors and successes, but unfortunately one test song did not provide enough data to get an accurate picture of how well the classifier performed.

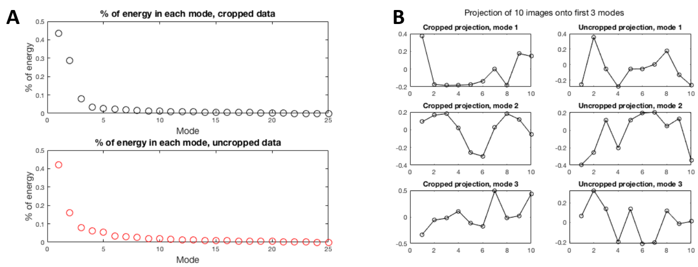

Figure 1: Results of SVD analysis on the Yale Faces dataset. Panel A shows the percent of the variance explained by each mode for cropped and uncropped images. Panel B shows the projections of ten images in each dataset onto the first three modes.

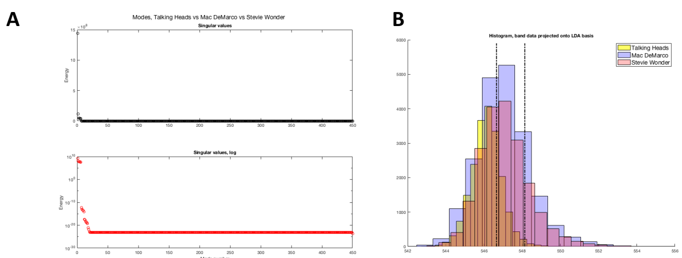

Figure 2: Results of SVD and LDA analysis on three bands across different genres (Talking Heads, New Wave; Mac DeMarco, Alternative; and Stevie Wonder, Soul). Panel A shows the percent of the variance captured by each mode after SVD. Panel B shows the histogram of band distributions, with thresholds between bands marked with dotted black lines.

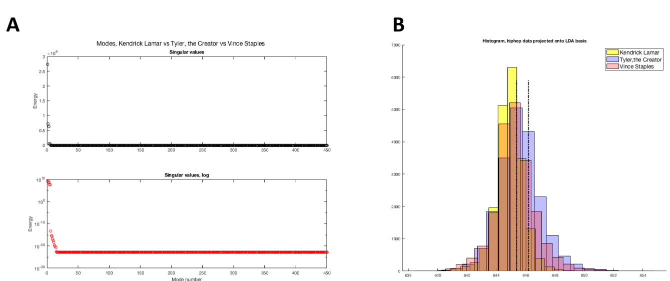

Figure 3: Results of SVD and LDA analysis on three bands within the same genre, Hip Hop (Kendrick Lamar, Tyler the Creator, and Vince Staples). Panel A shows the percent of the variance captured by each mode after SVD. Panel B shows the histogram of artist distributions, with thresholds between artists marked with dotted black lines.

5 Summary and Conclusions

Machine learning is a powerful tool that builds upon many foundational elements of data analysis. With increased dataset sizes and better time management, the author is confident that algorithms could be constructed that would perform decently okay at classifying facial expression images and songs across and within genres. Additionally, had the author realized earlier on in her code-writing that functions exist, she might have been able to submit code that does more in significantly fewer lines. Alas, the author was feeling overconfident in her increased MATLAB skills following this class, only to learn that there is still so much to learn.

Wavelet decomposition, SVD, and LDA all essentially decompose and transform data, but the power with which they do so should not be underestimated. As it become easier and easier to collect and store enormous datasets, these mathetmatical methods will only gain popularity among data scientists and researchers alike.

6 References

1. Kutz, J.N. (2013). Data-Driven Modeling Scientific Computation: Methods for Complex Systems and Big Data. Chapter 15. Oxford University Press. 1st. Ed.

Appendix A MATLAB Functions

Notable functions used in this analysis:

- `dir(path)` Produces a structure array listing the files in the given folder.
- `uigetfile` Opens dialogue box to choose and import data files from local drive.
- `imread` Reads in image data.
- `size(X)` Determines the size of X, or how many rows and columns X has.
- `reshape` Reshapes an array based on inputed size.
- `mean(X)` Determines the mean of X.
- `diag(X)` Returns the values that are on the diagonal of the matrix X.
- `[U S V]=svd(X)` Takes the singular value decomposition of X and returns the corresponding U, S, and V matrices.
- `audioread` Reads in audio data.
- `length(X)` Returns the length of the vector X.
- `plot(X,Y)` Plots 2D graph of X vs. Y.
- `for` Iterative statement that loops over given values.
- `eig(X)` Returns the eigenvalues of the input matrix X.
- `max(X)` Finds the maximum value in X.
- `fft(X)` Performs the fast Fourier transform on X.
- `exp(X)` Returns the exponential e for X.
- `abs(X)` Returns the absolute values of values in X.

Appendix B MATLAB Code

```
%% AMATH Homework 4

%% Yale Faces
close all; clear all; clc;

%Set path
croppedPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/CroppedYale/CroppedTrain/*.pgm';
uncroppedPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/UncroppedYale/.*pgm';
croppedFiles=dir(croppedPath);

%Load 25 cropped images to train with
for i=1:25
    file=uigetfile(croppedPath);
    croppedTrainSet(i).image=imread(file);
    croppedTrainSet(i).name=file;
end

imagedata=[];
for i=1:length(croppedTrainSet)
    imagedata(:,:,i)=croppedTrainSet(i).image;
end

%Reshape imagedata so that each column is a new image
size_data=size(imagedata);
imagedata=reshape(imagedata,[size_data(1)*size_data(2),size_data(3)]);


%SVD analysis
[m,n]=size(imagedata);
mn=mean(imagedata,2);
imagedata=imagedata-repmat(mn,1,n);
imagedata(isnan(imagedata))=0; imagedata(isinf(imagedata))=0;
[uc,sc,vc]=svd([imagedata],0); %SVD
lambda=diag(sc).^2; %diagonal variances
projc=uc'*imagedata;

%Load in 10 uncropped images
uncroppedFiles=dir(uncroppedPath);

%Load 25 cropped images to train with
for i=1:25
    file=uigetfile(uncroppedPath);
    uncroppedTrainSet(i).image=imread(file);
    uncroppedTrainSet(i).name=file;
end

ucimage=[];
for i=1:length(uncroppedTrainSet)
    ucimage(:,:,i)=uncroppedTrainSet(i).image;
end

%Reshape ucimg so that each column is a new image
```

```
size_ucimg=size(ucimage);
ucimage=reshape(ucimage,[size_ucimg(1)*size_ucimg(2),size_ucimg(3)]);

%SVD analysis, uncropped images
ucimage=double(ucimage);
[m,n]=size(ucimage);
mn=mean(ucimage,2);
ucimage=ucimage-repmat(mn,1,n);
[uuc,suc,vuc]=svd([ucimage],0); %SVD
lambdauc=diag(suc).^2; %diagonal variances
projuc=uuc'*ucimage;

figure(1)
subplot(2,1,1); plot(lambda/sum(lambda),'ok','MarkerSize',10); title('% of energy in each mode, cropped
ylabel('% of energy','FontSize',[14]); xlabel('Mode','FontSize',[14]);
subplot(2,1,2);plot(lambdauc/sum(lambdauc),'or','MarkerSize',10);title('% of energy in each mode, uncrop
ylabel('% of energy','FontSize',[14]); xlabel('Mode','FontSize',[14]);

figure(2)
for j=1:3
    subplot(3,2,2*j-1); plot(1:10,vc(1:10,j),'ko-'); title(sprintf('Cropped projection, mode %d',j));
    subplot(3,2,2*j); plot(1:10,vuc(1:10,j),'ko-'); title(sprintf('Uncropped projection, mode %d',j));
end
sgtitle('Projection of 10 images onto first 3 modes');

%% Music Genres
close all; clear all; clc;
%Load in data
newWaveTrainPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/Music_NewWave/CutNewWave/*.mp3'; nwT
demarcoTrainPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/Music_Mac/CutMac/*.mp3'; macTrainFil
soulTrainPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/Music_Soul/CutSoul/*.mp3'; soulTrainFil

for i=1:length(nwTrainFiles)
    [file, path]=uigetfile(newWaveTrainPath);
    newWave(i).audio=audioread(file);
    newWave(i).name=file;
end

for i=1:length(macTrainFiles)
    [file, path]=uigetfile(demarcoTrainPath);
    demarco(i).audio=audioread(file);
    demarco(i).name=file;
end

for i=1:length(soulTrainFiles)
    [file, path]=uigetfile(soulTrainPath);
    soul(i).audio=audioread(file);
    soul(i).name=file;
end
%% Section I: Individual bands, different genres

%% New Wave (Talking Heads) - Load audio data and compute spectrograms
nw1aud=newWave(1).audio; nw2aud=newWave(2).audio; nw3aud=newWave(3).audio; %get audio data out of struc
nw1t=6; nw2t=7; nw3t=6; %length of clip, in seconds
```

```
nw1fs=length(nw1aud)/nw1t; nw2fs=length(nw2aud)/nw2t; nw3fs=length(nw3aud)/nw3t; %determine Fs

knw1=(0:length(nw1aud)-1)*((nw1fs/2)/length(nw1aud)); knw1s=fftshift(knw1);
knw2=(0:length(nw2aud)-1)*((nw2fs/2)/length(nw2aud)); knw2s=fftshift(knw2);
knw3=(0:length(nw3aud)-1)*((nw3fs/2)/length(nw3aud)); knw3s=fftshift(knw3);

min_sizenw=min([size(nw1aud,1) size(nw2aud,1) size(nw3aud,1)]);
nw1aud=nw1aud(1:min_sizenw); nw2aud=nw2aud(1:min_sizenw); nw3aud=nw3aud(1:min_sizenw);
nw1f=fft(nw1aud); nw2f=fft(nw2aud); nw3f=fft(nw3aud);
knw1s=knw1s(1:min_sizenw); knw2s=knw2s(1:min_sizenw); knw3s=knw3s(1:min_sizenw);

figure(1) %Frequency data, new wave
subplot(3,1,1); plot(knw1s,abs(fftshift(nw1f)));
subplot(3,1,2); plot(knw2s,abs(fftshift(nw2f)));
subplot(3,1,3); plot(knw3s,abs(fftshift(nw3f)));

nw1_spec=[];
t1slide=[0:0.1:nw1t];
for i=1:length(t1slide)
    g=exp(-25*(nw1t-t1slide(i)).^2);
    nw1g=g.*nw1aud;
    nw1gt=fft(nw1g);
    nw1_spec=[nw1_spec; abs(nw1gt)];
end

nw2_spec=[];
t2slide=[0:0.1:nw2t];
for i=1:length(t2slide);
    g=exp(-25*(nw2t-t2slide(i)).^2);
    nw2g=g.*nw2aud;
    nw2gt=fft(nw2g);
    nw2_spec=[nw2_spec; abs(nw2gt)];
end

nw3_spec=[];
t3slide=[0:0.1:nw3t];
for i=1:length(t3slide);
    g=exp(-25*(nw3t-t3slide(i)).^2);
    nw3g=g.*nw3aud;
    nw3gt=fft(nw3g);
    nw3_spec=[nw3_spec; abs(nw3gt)];
end

nw_matrix=[nw1_spec;nw2_spec;nw3_spec];

%% Modern alternative (Mac DeMarco) - Load audio data and compute spectrograms
dm1aud=demarco(1).audio; dm2aud=demarco(2).audio; dm3aud=demarco(3).audio; %get audio data out of struc
dm1t=5; dm2t=6; dm3t=7; %length of clip, in seconds
dm1fs=length(dm1aud)/dm1t; dm2fs=length(dm2aud)/dm2t; dm3fs=length(dm3aud)/dm3t; %determine Fs

kdm1=(0:length(dm1aud)-1)*((dm1fs/2)/length(dm1aud)); kdm1s=fftshift(kdm1);
kdm2=(0:length(dm2aud)-1)*((dm2fs/2)/length(dm2aud)); kdm2s=fftshift(kdm2);
kdm3=(0:length(dm3aud)-1)*((dm3fs/2)/length(dm3aud)); kdm3s=fftshift(kdm3);
```

```
min_sizedm=min([size(dm1aud,1) size(dm2aud,1) size(dm3aud,1)]);
dm1aud=dm1aud(1:min_sizedm); dm2aud=dm2aud(1:min_sizedm); dm3aud=dm3aud(1:min_sizedm);
dm1f=fft(dm1aud); dm2f=fft(dm2aud); dm3f=fft(dm3aud);
kdm1s=kdm1s(1:min_sizedm); kdm2s=kdm2s(1:min_sizedm); kdm3s=kdm3s(1:min_sizedm);

figure(1) %Frequency data, new wave
subplot(3,1,1); plot(kdm1s,abs(fftshift(dm1f)));
subplot(3,1,2); plot(kdm2s,abs(fftshift(dm2f)));
subplot(3,1,3); plot(kdm3s,abs(fftshift(dm3f)));
title('Mac Demarco - FFT');

dm1_spec=[];
dm1slide=[0:0.1:dm1t];
for i=1:length(dm1slide)
    g=exp(-25*(dm1t-dm1slide(i)).^2);
    dm1g=g.*dm1aud;
    dm1gt=fft(dm1g);
    dm1_spec=[dm1_spec; abs(dm1gt)];
end

dm2_spec=[];
dm2slide=[0:0.1:dm2t];
for i=1:length(dm2slide);
    g=exp(-25*(dm2t-dm2slide(i)).^2);
    dm2g=g.*dm2aud;
    dm2gt=fft(dm2g);
    dm2_spec=[dm2_spec; abs(dm2gt)];
end

dm3_spec=[];
dm3slide=[0:0.1:dm3t];
for i=1:length(dm3slide);
    g=exp(-25*(dm3t-dm3slide(i)).^2);
    dm3g=g.*dm3aud;
    dm3gt=fft(dm3g);
    dm3_spec=[dm3_spec; abs(dm3gt)];
end

dm_matrix=[dm1_spec;dm2_spec;dm3_spec];

%% Soul (Stevie Wonder) - Load audio data and compute spectrograms
sw1aud=soul(7).audio; sw2aud=soul(8).audio; sw3aud=soul(9).audio; %get audio data out of struct
sw1t=5; sw2t=5; sw3t=9; %length of clip, in seconds
sw1fs=length(sw1aud)/sw1t; sw2fs=length(sw2aud)/sw2t; sw3fs=length(sw3aud)/sw3t; %determine Fs

ksw1=(0:length(sw1aud)-1)*((sw1fs/2)/length(sw1aud)); ksw1s=fftshift(ksw1);
ksw2=(0:length(sw2aud)-1)*((sw2fs/2)/length(sw2aud)); ksw2s=fftshift(ksw2);
ksw3=(0:length(sw3aud)-1)*((sw3fs/2)/length(sw3aud)); ksw3s=fftshift(ksw3);

min_sizesw=min([size(sw1aud,1) size(sw2aud,1) size(sw3aud,1)]);
sw1aud=sw1aud(1:min_sizesw); sw2aud=sw2aud(1:min_sizesw); sw3aud=sw3aud(1:min_sizesw);
sw1f=fft(sw1aud); sw2f=fft(sw2aud); sw3f=fft(sw3aud);
ksw1s=ksw1s(1:min_sizesw); ksw2s=ksw2s(1:min_sizesw); ksw3s=ksw3s(1:min_sizesw);
```

```
figure(1) %Frequency data
subplot(3,1,1); plot(ksw1s,abs(fftshift(sw1f)));
subplot(3,1,2); plot(ksw2s,abs(fftshift(sw2f)));
subplot(3,1,3); plot(ksw3s,abs(fftshift(sw3f)));
title('Stevie Wonder - FFT');

sw1_spec=[];
sw1slide=[0:0.1:sw1t];
for i=1:length(sw1slide)
    g=exp(-25*(sw1t-sw1slide(i)).^2);
    sw1g=g.*sw1aud;
    sw1gt=fft(sw1g);
    sw1_spec=[sw1_spec; abs(sw1gt)];
end

sw2_spec=[];
sw2slide=[0:0.1:sw2t];
for i=1:length(sw2slide);
    g=exp(-25*(sw2t-sw2slide(i)).^2);
    sw2g=g.*sw2aud;
    sw2gt=fft(sw2g);
    sw2_spec=[sw2_spec; abs(sw2gt)];
end

sw3_spec=[];
sw3slide=[0:0.1:sw3t];
for i=1:length(sw3slide);
    g=exp(-25*(sw3t-sw3slide(i)).^2);
    sw3g=g.*sw3aud;
    sw3gt=fft(sw3g);
    sw3_spec=[sw3_spec; abs(sw3gt)];
end

sw_matrix=[sw1_spec;sw2_spec;sw3_spec];


%% Talking Heads vs Mac Demarco vs Stevie Wonder - SVD and LDA
band_sizes=[size(nw_matrix); size(dm_matrix); size(sw_matrix)];

%resample to take first 150 rows and 20000 columns of each matrix
nw_matrix=nw_matrix(1:150,1:20000); dm_matrix=dm_matrix(1:150,1:20000); sw_matrix=sw_matrix(1:150,1:2000

%SVD
band_matrix=[nw_matrix; dm_matrix; sw_matrix];
[m,n]=size(band_matrix);
mn=mean(band_matrix,2);
band_matrix=band_matrix-repmat(mn,1,n);
[u_band,s_band,v_band]=svd(band_matrix/sqrt(band_matrix-n)); %SVD
lambda_band=diag(s_band).^2; %diagonal variances
proj_band=u_band'*band_matrix;

figure(2)
subplot(2,1,1); plot(lambda_band,'ko'); title('Singular values'); ylabel('Energy');
subplot(2,1,2); semilogy(lambda_band,'ro'); title('Singular values, log'); ylabel('Energy');
```

```
xlabel('Mode number');
sgtitle('Modes, Talking Heads vs Mac DeMarco vs Stevie Wonder')

n_nw=length(nw_matrix(1,:)); n_dm=length(dm_matrix(1,:)); n_sw=length(sw_matrix(1,:));
feature=20; %20 PCs to classify band features

m_nw=mean(nw_matrix,2); m_dm=mean(dm_matrix,2); m_sw=mean(sw_matrix,2); %means
Wc=0; %within class variance
for i=1:n_nw
    Wc=Wc+(nw_matrix(:,i)-m_nw)*(nw_matrix(:,i)-m_nw)';
end
for i=1:n_dm
    Wc=Wc+(dm_matrix(:,i)-m_dm)*(dm_matrix(:,i)-m_dm)';
end
for i=1:n_sw
    Wc=Wc+(sw_matrix(:,i)-m_sw)*(sw_matrix(:,i)-m_sw)';
end

%LDA
Bc=(m_nw-m_dm-m_sw)*(m_nw-m_dm-m_sw)';
[V2,D]=eig(Wc,Bc); [lambda,idx]=max(abs(diag(D))); w=V2(:,idx); w=w/norm(w,2);
v_nw=w'*nw_matrix; v_dm=w'*dm_matrix; v_sw=w'*sw_matrix;
result=[v_nw,v_dm,v_sw];

if mean(v_nw)>mean(v_dm)>mean(v_sw)
    w=-w;
    v_nw=-v_nw;
    v_dm=-v_dm;
    v_sw=-v_sw;
end

sort_nw=sort(v_nw); sort_dm=sort(v_dm); sort_sw=sort(v_sw);
t1=length(sort_nw); t2=1;
while sort_nw(t1)>sort_dm(t2)
    t1=t1-1;
    t2=t2+1;
end
threshold_nwdm=(sort_nw(t1)+sort_dm(t2))/2;
while sort_dm(t1)>sort_sw(t2)
    t1=t1-1;
    t2=t2+1;
end
threshold_dmsw=(sort_dm(t1)+sort_sw(t2))/2;

%Histogram and thresholds
figure(3)
hold on
histogram(abs(log(sort_nw)),20,'FaceColor','yellow');
histogram(abs(log(sort_dm)),20,'FaceColor','blue','FaceAlpha',0.25);
histogram(abs(log(sort_sw)),20,'FaceColor','red','FaceAlpha',0.25);
plot([abs(log(threshold_nwdm)) abs(log(threshold_nwdm))],[0 5900],'-.k','LineWidth',2);
plot([abs(log(threshold_dmsw)) abs(log(threshold_dmsw))],[0 5900],'-.k','LineWidth',2);
xlim([542 556]); legend('Talking Heads','Mac DeMarco', 'Stevie Wonder','FontSize',14);
title('Histogram, band data projected onto LDA basis')
```

```
%% Determine accuracy of TH vs MD vs SW training with additional 5s clips
testPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/TestMusic/*.mp3';
testFiles=dir(testPath);

for i=1:length(testFiles)
    file=uigetfile(testPath);
    testData(i).audio=audioread(file);
    testData(i).name=file;
end

mactestaud=testData(1).audio; mactestt=8; mactestfs=length(mactestaud)/mactestt;

mactestspec=[];
macslide=0:0.1:mactestt;
for i=1:length(macslide)
    g=exp(-25*(mactestt-macslide(i)).^2);
    mactestg=g.*mactestaud;
    mactestgt=fft(mactestg);
    mactestspec=[mactestspec; abs(mactestgt)];
end

hiddenlabels=[1];
mactestspec=mactestspec(1:length(u_band));
testMac=u_band'.*mactestspec;
testMac=testMac(1:length(w));
pval=w'.*testMac;

ResVec=(pval>threshold_nwdm);
errNum=sum(abs(ResVec-hiddenlabels));
sucRate=1-errNum/1;
%% Section II: Bands within genre - Hip Hop
hipHopTrainPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/Music_HipHop/CutHipHop/*.mp3'; hipHopT

for i=1:length(hipHopTrainFiles)
    [file, path]=uigetfile(hipHopTrainPath);
    hiphopTrain(i).audio=audioread(file);
    hiphopTrain(i).name=file;
end

%% Hip Hop - Kendrick Lamar - Load audio data and compute spectrograms
kendrick1aud=hiphopTrain(1).audio; kendrick2aud=hiphopTrain(2).audio; kendrick3aud=hiphopTrain(3).audio
kendrick1t=6; kendrick2t=10; kendrick3t=6; %length of clip, in seconds
kendrick1fs=length(kendrick1aud)/kendrick1t; kendrick2fs=length(kendrick2aud)/kendrick2t; kendrick3fs=le

kkendrick1=(0:length(kendrick1aud)-1)*((kendrick1fs/2)/length(kendrick1aud)); kkendrick1s=fftshift(kkend
kkendrick2=(0:length(kendrick2aud)-1)*((kendrick2fs/2)/length(kendrick2aud)); kkendrick2s=fftshift(kkend
kkendrick3=(0:length(kendrick3aud)-1)*((kendrick3fs/2)/length(kendrick3aud)); kkendrick3s=fftshift(kkend

min_sizekendrick=min([size(kendrick1aud,1) size(kendrick2aud,1) size(kendrick3aud,1)]);
kendrick1aud=kendrick1aud(1:min_sizekendrick); kendrick2aud=kendrick2aud(1:min_sizekendrick); kendrick3a
kendrick1f=fft(kendrick1aud); kendrick2f=fft(kendrick2aud); kendrick3f=fft(kendrick3aud);
kkendrick1s=kkendrick1s(1:min_sizekendrick); kkendrick2s=kkendrick2s(1:min_sizekendrick); kkendrick3s=kk
```

```
kendrick1_spec=[];
kendrick1slide=[0:0.1:kendrick1t];
for i=1:length(kendrick1slide)
    g=exp(-25*(kendrick1t-kendrick1slide(i)).^2);
    kendrick1g=g.*kendrick1aud;
    kendrick1gt=fft(kendrick1g);
    kendrick1_spec=[kendrick1_spec; abs(kendrick1gt)];
end

kendrick2_spec=[];
kendrick2slide=[0:0.1:kendrick2t];
for i=1:length(kendrick2slide);
    g=exp(-25*(kendrick2t-kendrick2slide(i)).^2);
    kendrick2g=g.*kendrick2aud;
    kendrick2gt=fft(kendrick2g);
    kendrick2_spec=[kendrick2_spec; abs(kendrick2gt)];
end

kendrick3_spec=[];
kendrick3slide=[0:0.1:kendrick3t];
for i=1:length(kendrick3slide);
    g=exp(-25*(kendrick3t-kendrick3slide(i)).^2);
    kendrick3g=g.*kendrick3aud;
    kendrick3gt=fft(kendrick3g);
    kendrick3_spec=[kendrick3_spec; abs(kendrick3gt)];
end

kendrick_matrix=[kendrick1_spec;kendrick2_spec;kendrick3_spec];
%% Hip Hop - Tyler, the Creator - Load audio data and compute spectrograms
tyler1aud=hiphopTrain(4).audio; tyler2aud=hiphopTrain(5).audio; tyler3aud=hiphopTrain(6).audio; %get au
tyler1t=6; tyler2t=6; tyler3t=10; %length of clip, in seconds
tyler1fs=length(tyler1aud)/tyler1t; tyler2fs=length(tyler2aud)/tyler2t; tyler3fs=length(tyler3aud)/tyle

ktyler1=(0:length(tyler1aud)-1)*((tyler1fs/2)/length(tyler1aud)); ktyler1s=fftshift(ktyler1);
ktyler2=(0:length(tyler2aud)-1)*((tyler2fs/2)/length(tyler2aud)); ktyler2s=fftshift(ktyler2);
ktyler3=(0:length(tyler3aud)-1)*((tyler3fs/2)/length(tyler3aud)); ktyler3s=fftshift(ktyler3);

min_sizetyler=min([size(tyler1aud,1) size(tyler2aud,1) size(tyler3aud,1)]);
tyler1aud=tyler1aud(1:min_sizetyler); tyler2aud=tyler2aud(1:min_sizetyler); tyler3aud=tyler3aud(1:min_s
tyler1f=fft(tyler1aud); tyler2f=fft(tyler2aud); tyler3f=fft(tyler3aud);
ktyler1s=ktyler1s(1:min_sizetyler); ktyler2s=ktyler2s(1:min_sizetyler); ktyler3s=ktyler3s(1:min_sizetyl

tyler1_spec=[];
tyler1slide=[0:0.1:tyler1t];
for i=1:length(tyler1slide)
    g=exp(-25*(tyler1t-tyler1slide(i)).^2);
    tyler1g=g.*tyler1aud;
    tyler1gt=fft(tyler1g);
    tyler1_spec=[tyler1_spec; abs(tyler1gt)];
end

tyler2_spec=[];
tyler2slide=[0:0.1:tyler2t];
for i=1:length(tyler2slide);
```

```
    g=exp(-25*(tyler2t-tyler2slide(i)).^2);
    tyler2g=g.*tyler2aud;
    tyler2gt=fft(tyler2g);
    tyler2_spec=[tyler2_spec; abs(tyler2gt)];
end

tyler3_spec=[];
tyler3slide=[0:0.1:tyler3t];
for i=1:length(tyler3slide);
    g=exp(-25*(tyler3t-tyler3slide(i)).^2);
    tyler3g=g.*tyler3aud;
    tyler3gt=fft(tyler3g);
    tyler3_spec=[tyler3_spec; abs(tyler3gt)];
end

tyler_matrix=[tyler1_spec;tyler2_spec;tyler3_spec];

%% Hip Hop - Vince Staples - Load audio data and compute spectrograms
vince1aud=hiphopTrain(7).audio; vince2aud=hiphopTrain(8).audio; vince3aud=hiphopTrain(9).audio; %get au
vince1t=6; vince2t=5; vince3t=7; %length of clip, in seconds
vince1fs=length(vince1aud)/vince1t; vince2fs=length(vince2aud)/vince2t; vince3fs=length(vince3aud)/vinc

kvince1=(0:length(vince1aud)-1)*((vince1fs/2)/length(vince1aud)); kvince1s=fftshift(kvince1);
kvince2=(0:length(vince2aud)-1)*((vince2fs/2)/length(vince2aud)); kvince2s=fftshift(kvince2);
kvince3=(0:length(vince3aud)-1)*((vince3fs/2)/length(vince3aud)); kvince3s=fftshift(kvince3);

min_sizevince=min([size(vince1aud,1) size(vince2aud,1) size(vince3aud,1)]);
vince1aud=vince1aud(1:min_sizevince); vince2aud=vince2aud(1:min_sizevince); vince3aud=vince3aud(1:min_s
vince1f=fft(vince1aud); vince2f=fft(vince2aud); vince3f=fft(vince3aud);
kvince1s=kvince1s(1:min_sizevince); kvince2s=kvince2s(1:min_sizevince); kvince3s=kvince3s(1:min_sizevin

vince1_spec=[];
vince1slide=[0:0.1:vince1t];
for i=1:length(vince1slide)
    g=exp(-25*(vince1t-vince1slide(i)).^2);
    vince1g=g.*vince1aud;
    vince1gt=fft(vince1g);
    vince1_spec=[vince1_spec; abs(vince1gt)];
end

vince2_spec=[];
vince2slide=[0:0.1:vince2t];
for i=1:length(vince2slide);
    g=exp(-25*(vince2t-vince2slide(i)).^2);
    vince2g=g.*vince2aud;
    vince2gt=fft(vince2g);
    vince2_spec=[vince2_spec; abs(vince2gt)];
end

vince3_spec=[];
vince3slide=[0:0.1:vince3t];
for i=1:length(vince3slide);
    g=exp(-25*(vince3t-vince3slide(i)).^2);
    vince3g=g.*vince3aud;
```

```
    vince3gt=fft(vince3g);
    vince3_spec=[vince3_spec; abs(vince3gt)];
end

vince_matrix=[vince1_spec;vince2_spec;vince3_spec];

%% Hip Hop - SVD and LDA
hiphop_sizes=[size(kendrick_matrix); size(tyler_matrix); size(vince_matrix)];

%resample to take first 150 rows and 20000 columns of each matrix
kendrick_matrix=kendrick_matrix(1:150,1:20000); tyler_matrix=tyler_matrix(1:150,1:20000); vince_matrix=

%SVD
hiphop_matrix=[kendrick_matrix; tyler_matrix; vince_matrix];
[m,n]=size(hiphop_matrix);
mn=mean(hiphop_matrix,2);
hiphop_matrix=hiphop_matrix-repmat(mn,1,n);
[u_hiphop,s_hiphop,v_hiphop]=svd(hiphop_matrix/sqrt(hiphop_matrix-n)); %SVD
lambda_hiphop=diag(s_hiphop).^2; %diagonal variances
proj_hiphop=u_hiphop'*hiphop_matrix;

figure(2)
subplot(2,1,1); plot(lambda_hiphop,'ko'); title('Singular values'); ylabel('Energy');
subplot(2,1,2); semilogy(lambda_hiphop,'ro'); title('Singular values, log'); ylabel('Energy');
xlabel('Mode number');
sgtitle('Modes, Kendrick Lamar vs Tyler, the Creator vs Vince Staples')

n_kendrick=length(kendrick_matrix(1,:)); n_tyler=length(tyler_matrix(1,:)); n_vince=length(vince_matrix
feature=10; %20 PCs to classify hiphop features

m_kendrick=mean(kendrick_matrix,2); m_tyler=mean(tyler_matrix,2); m_vince=mean(vince_matrix,2); %means
Wc=0; %within class variance
for i=1:n_kendrick
    Wc=Wc+(kendrick_matrix(:,i)-m_kendrick)*(kendrick_matrix(:,i)-m_kendrick)';
end
for i=1:n_tyler
    Wc=Wc+(tyler_matrix(:,i)-m_tyler)*(tyler_matrix(:,i)-m_tyler)';
end
for i=1:n_vince
    Wc=Wc+(vince_matrix(:,i)-m_vince)*(vince_matrix(:,i)-m_vince)';
end

%LDA
Bc=(m_kendrick-m_tyler-m_vince)*(m_kendrick-m_tyler-m_vince)';
[V2,D]=eig(Wc,Bc); [lambda,idx]=max(abs(diag(D))); w=V2(:,idx); w=w/norm(w,2);
v_kendrick=w'*kendrick_matrix; v_tyler=w'*tyler_matrix; v_vince=w'*vince_matrix;
result=[v_kendrick,v_tyler,v_vince];

if mean(v_kendrick)>mean(v_tyler)>mean(v_vince)
    w=-w;
    v_kendrick=-v_kendrick;
    v_tyler=-v_tyler;
    v_vince=-v_vince;
end
```

```
sort_kendrick=sort(v_kendrick); sort_tyler=sort(v_tyler); sort_vince=sort(v_vince);
t1=length(sort_kendrick); t2=1;
while sort_kendrick(t1)>sort_tyler(t2)
    t1=t1-1;
    t2=t2+1;
end
threshold_kendricktyler=(sort_kendrick(t1)+sort_tyler(t2))/2;
while sort_tyler(t1)>sort_vince(t2)
    t1=t1-1;
    t2=t2+1;
end
threshold_tylervince=(sort_tyler(t1)+sort_vince(t2))/2;

%Histogram and thresholds
figure(3)
hold on
histogram(abs(log(sort_kendrick)),20,'FaceColor','yellow');
histogram(abs(log(sort_tyler)),20,'FaceColor','blue','FaceAlpha',0.25);
histogram(abs(log(sort_vince)),20,'FaceColor','red','FaceAlpha',0.25);
plot([abs(log(threshold_kendricktyler)) abs(log(threshold_kendricktyler))],[0 5900],'-.k','LineWidth',2
plot([abs(log(threshold_tylervince)) abs(log(threshold_tylervince))],[0 5900],'-.k','LineWidth',2);
legend('Kendrick Lamar','Tyler,the Creator', 'Vince Staples','FontSize',14);
title('Histogram, hiphop data projected onto LDA basis')

%% Hip Hop - Determine accuracy of classification


%% Section III: Genre classification - Hip Hop vs Soul vs "Sad Dad" Alternative
sadDadTrainPath='/Users/kristendrummey/Desktop/AMATH582/Homework4/Music_SadDad/CutSadDad/*.mp3'; sadTra

for i=1:length(sadFiles)
    [file, path]=uigetfile(sadDadPath);
    sadDadTrain(i).audio=audioread(file);
    sadDadTrain(i).name=file;
end

%% Sad Dad - Load audio data and compute spectrograms (Bon Iver)
boniver1aud=sadDadTrain(1).audio; boniver2aud=sadDadTrain(2).audio; boniver3aud=sadDadTrain(3).audio; %
boniver1t=6; boniver2t=7; boniver3t=5; %length of clip, in seconds
boniver1fs=length(boniver1aud)/boniver1t; boniver2fs=length(boniver2aud)/boniver2t; boniver3fs=length(b

kboniver1=(0:length(boniver1aud)-1)*((boniver1fs/2)/length(boniver1aud)); kboniver1s=fftshift(kboniver1
kboniver2=(0:length(boniver2aud)-1)*((boniver2fs/2)/length(boniver2aud)); kboniver2s=fftshift(kboniver2
kboniver3=(0:length(boniver3aud)-1)*((boniver3fs/2)/length(boniver3aud)); kboniver3s=fftshift(kboniver3

min_sizeboniver=min([size(boniver1aud,1) size(boniver2aud,1) size(boniver3aud,1)]);
boniver1aud=boniver1aud(1:min_sizeboniver); boniver2aud=boniver2aud(1:min_sizeboniver); boniver3aud=bon
boniver1f=fft(boniver1aud); boniver2f=fft(boniver2aud); boniver3f=fft(boniver3aud);
kboniver1s=kboniver1s(1:min_sizeboniver); kboniver2s=kboniver2s(1:min_sizeboniver); kboniver3s=kboniver

boniver1_spec=[];
boniver1slide=[0:0.1:boniver1t];
for i=1:length(boniver1slide)
```

```
    g=exp(-25*(boniver1t-boniver1slide(i)).^2);
    boniver1g=g.*boniver1aud;
    boniver1gt=fft(boniver1g);
    boniver1_spec=[boniver1_spec; abs(boniver1gt)];
end

boniver2_spec=[];
boniver2slide=[0:0.1:boniver2t];
for i=1:length(boniver2slide);
    g=exp(-25*(boniver2t-boniver2slide(i)).^2);
    boniver2g=g.*boniver2aud;
    boniver2gt=fft(boniver2g);
    boniver2_spec=[boniver2_spec; abs(boniver2gt)];
end

boniver3_spec=[];
boniver3slide=[0:0.1:boniver3t];
for i=1:length(boniver3slide);
    g=exp(-25*(boniver3t-boniver3slide(i)).^2);
    boniver3g=g.*boniver3aud;
    boniver3gt=fft(boniver3g);
    boniver3_spec=[boniver3_spec; abs(boniver3gt)];
end

boniver_matrix=[boniver1_spec;boniver2_spec;boniver3_spec];
%% Sad Dad - Load audio data and compute spectrograms (Local Natives)
locnat1aud=sadDadTrain(4).audio; locnat2aud=sadDadTrain(5).audio; locnat3aud=sadDadTrain(6).audio; %get
locnat1t=6; locnat2t=8; locnat3t=6; %length of clip, in seconds
locnat1fs=length(locnat1aud)/locnat1t; locnat2fs=length(locnat2aud)/locnat2t; locnat3fs=length(locnat3au

klocnat1=(0:length(locnat1aud)-1)*((locnat1fs/2)/length(locnat1aud)); klocnat1s=fftshift(klocnat1);
klocnat2=(0:length(locnat2aud)-1)*((locnat2fs/2)/length(locnat2aud)); klocnat2s=fftshift(klocnat2);
klocnat3=(0:length(locnat3aud)-1)*((locnat3fs/2)/length(locnat3aud)); klocnat3s=fftshift(klocnat3);

min_sizelocnat=min([size(locnat1aud,1) size(locnat2aud,1) size(locnat3aud,1)]);
locnat1aud=locnat1aud(1:min_sizelocnat); locnat2aud=locnat2aud(1:min_sizelocnat); locnat3aud=locnat3aud
locnat1f=fft(locnat1aud); locnat2f=fft(locnat2aud); locnat3f=fft(locnat3aud);
klocnat1s=klocnat1s(1:min_sizelocnat); klocnat2s=klocnat2s(1:min_sizelocnat); klocnat3s=klocnat3s(1:min_

locnat1_spec=[];
locnat1slide=[0:0.1:locnat1t];
for i=1:length(locnat1slide)
    g=exp(-25*(locnat1t-locnat1slide(i)).^2);
    locnat1g=g.*locnat1aud;
    locnat1gt=fft(locnat1g);
    locnat1_spec=[locnat1_spec; abs(locnat1gt)];
end

locnat2_spec=[];
locnat2slide=[0:0.1:locnat2t];
for i=1:length(locnat2slide);
    g=exp(-25*(locnat2t-locnat2slide(i)).^2);
    locnat2g=g.*locnat2aud;
    locnat2gt=fft(locnat2g);
```

```
    locnat2_spec=[locnat2_spec; abs(locnat2gt)];
end

locnat3_spec=[];
locnat3slide=[0:0.1:locnat3t];
for i=1:length(locnat3slide);
    g=exp(-25*(locnat3t-locnat3slide(i)).^2);
    locnat3g=g.*locnat3aud;
    locnat3gt=fft(locnat3g);
    locnat3_spec=[locnat3_spec; abs(locnat3gt)];
end

locnat_matrix=[locnat1_spec;locnat2_spec;locnat3_spec];
%% Sad Dad - Load audio data and compute spectrograms (The National)
national1aud=sadDadTrain(7).audio; national2aud=sadDadTrain(8).audio; sadDad3aud=hiphopTrain(9).audio; %
national1t=6; national2t=5; national3t=6; %length of clip, in seconds
national1fs=length(national1aud)/national1t; national2fs=length(national2aud)/national2t; national3fs=le

knational1=(0:length(national1aud)-1)*((national1fs/2)/length(national1aud)); knational1s=fftshift(knati
knational2=(0:length(national2aud)-1)*((national2fs/2)/length(national2aud)); knational2s=fftshift(knati
knational3=(0:length(national3aud)-1)*((national3fs/2)/length(national3aud)); knational3s=fftshift(knati

min_sizenational=min([size(national1aud,1) size(national2aud,1) size(national3aud,1)]);
national1aud=national1aud(1:min_sizenational); national2aud=national2aud(1:min_sizenational); national3a
national1f=fft(national1aud); national2f=fft(national2aud); national3f=fft(national3aud);
knational1s=knational1s(1:min_sizenational); knational2s=knational2s(1:min_sizenational); knational3s=kn

national1_spec=[];
national1slide=[0:0.1:national1t];
for i=1:length(national1slide)
    g=exp(-25*(national1t-national1slide(i)).^2);
    national1g=g.*national1aud;
    national1gt=fft(national1g);
    national1_spec=[national1_spec; abs(national1gt)];
end

national2_spec=[];
national2slide=[0:0.1:national2t];
for i=1:length(national2slide);
    g=exp(-25*(national2t-national2slide(i)).^2);
    national2g=g.*national2aud;
    national2gt=fft(national2g);
    national2_spec=[national2_spec; abs(national2gt)];
end

national3_spec=[];
national3slide=[0:0.1:national3t];
for i=1:length(national3slide);
    g=exp(-25*(national3t-national3slide(i)).^2);
    national3g=g.*national3aud;
    national3gt=fft(national3g);
    national3_spec=[national3_spec; abs(national3gt)];
end
```

```
national_matrix=[national1_spec;national2_spec;national3_spec];
%% Soul - Load audio data and compute spectrograms (Earth, Wind, and Fire)
earth1aud=soulTrain(1).audio; earth2aud=soulTrain(2).audio; earth3aud=soulTrain(3).audio; %get audio da
earth1t=6; earth2t=6; earth3t=7; %length of clip, in seconds
earth1fs=length(earth1aud)/earth1t; earth2fs=length(earth2aud)/earth2t; earth3fs=length(earth3aud)/eart

kearth1=(0:length(earth1aud)-1)*((earth1fs/2)/length(earth1aud)); kearth1s=fftshift(kearth1);
kearth2=(0:length(earth2aud)-1)*((earth2fs/2)/length(earth2aud)); kearth2s=fftshift(kearth2);
kearth3=(0:length(earth3aud)-1)*((earth3fs/2)/length(earth3aud)); kearth3s=fftshift(kearth3);

min_sizeearth=min([size(earth1aud,1) size(earth2aud,1) size(earth3aud,1)]);
earth1aud=earth1aud(1:min_sizeearth); earth2aud=earth2aud(1:min_sizeearth); earth3aud=earth3aud(1:min_s
earth1f=fft(earth1aud); earth2f=fft(earth2aud); earth3f=fft(earth3aud);
kearth1s=kearth1s(1:min_sizeearth); kearth2s=kearth2s(1:min_sizeearth); kearth3s=kearth3s(1:min_sizeear

earth1_spec=[];
earth1slide=[0:0.1:earth1t];
for i=1:length(earth1slide)
    g=exp(-25*(earth1t-earth1slide(i)).^2);
    earth1g=g.*earth1aud;
    earth1gt=fft(earth1g);
    earth1_spec=[earth1_spec; abs(earth1gt)];
end

earth2_spec=[];
earth2slide=[0:0.1:earth2t];
for i=1:length(earth2slide);
    g=exp(-25*(earth2t-earth2slide(i)).^2);
    earth2g=g.*earth2aud;
    earth2gt=fft(earth2g);
    earth2_spec=[earth2_spec; abs(earth2gt)];
end

earth3_spec=[];
earth3slide=[0:0.1:earth3t];
for i=1:length(earth3slide);
    g=exp(-25*(earth3t-earth3slide(i)).^2);
    earth3g=g.*earth3aud;
    earth3gt=fft(earth3g);
    earth3_spec=[earth3_spec; abs(earth3gt)];
end

earth_matrix=[earth1_spec;earth2_spec;earth3_spec];
%% Soul - Load audio data and compute spectrograms (Marvin Gaye)
marvin1aud=soulTrain(4).audio; marvin2aud=soulTrain(5).audio; marvin3aud=soulTrain(6).audio; %get audio
marvin1t=7; marvin2t=6; marvin3t=8; %length of clip, in seconds
marvin1fs=length(marvin1aud)/marvin1t; marvin2fs=length(marvin2aud)/marvin2t; marvin3fs=length(marvin3a

kmarvin1=(0:length(marvin1aud)-1)*((marvin1fs/2)/length(marvin1aud)); kmarvin1s=fftshift(kmarvin1);
kmarvin2=(0:length(marvin2aud)-1)*((marvin2fs/2)/length(marvin2aud)); kmarvin2s=fftshift(kmarvin2);
kmarvin3=(0:length(marvin3aud)-1)*((marvin3fs/2)/length(marvin3aud)); kmarvin3s=fftshift(kmarvin3);

min_sizemarvin=min([size(marvin1aud,1) size(marvin2aud,1) size(marvin3aud,1)]);
marvin1aud=marvin1aud(1:min_sizemarvin); marvin2aud=marvin2aud(1:min_sizemarvin); marvin3aud=marvin3aud
```

```
marvin1f=fft(marvin1aud); marvin2f=fft(marvin2aud); marvin3f=fft(marvin3aud);
kmarvin1s=kmarvin1s(1:min_sizemarvin); kmarvin2s=kmarvin2s(1:min_sizemarvin); kmarvin3s=kmarvin3s(1:min_

marvin1_spec=[];
marvin1slide=[0:0.1:marvin1t];
for i=1:length(marvin1slide)
    g=exp(-25*(marvin1t-marvin1slide(i)).^2);
    marvin1g=g.*marvin1aud;
    marvin1gt=fft(marvin1g);
    marvin1_spec=[marvin1_spec; abs(marvin1gt)];
end

marvin2_spec=[];
marvin2slide=[0:0.1:marvin2t];
for i=1:length(marvin2slide);
    g=exp(-25*(marvin2t-marvin2slide(i)).^2);
    marvin2g=g.*marvin2aud;
    marvin2gt=fft(marvin2g);
    marvin2_spec=[marvin2_spec; abs(marvin2gt)];
end

marvin3_spec=[];
marvin3slide=[0:0.1:marvin3t];
for i=1:length(marvin3slide);
    g=exp(-25*(marvin3t-marvin3slide(i)).^2);
    marvin3g=g.*marvin3aud;
    marvin3gt=fft(marvin3g);
    marvin3_spec=[marvin3_spec; abs(marvin3gt)];
end

marvin_matrix=[marvin1_spec;marvin2_spec;marvin3_spec];
%% Genre - SVD and LDA

%% Genre - Determine accuracy of classification
```