AMATH 482
Computational Methods for Data Analysis

Zach Shaw

Winter 2019

Homework 4

Due: March 8, 2019.

March 7, 2019

Abstract

In this homework, I analyzed the deconstruction of facial images using SVD and PCA. We had a cropped data set vs. and uncropped data set. I preformed PCA on both to determine how it behaves on an ideal vs. a noisy data set. The next part of the homework asked me to train an algorithm to classify music. I investigated 3 different cases. After preforming FFT on the samples, I achieved extremely high accuracy using 3 different classifiers.

1 Introduction and Overview

This homework is an introduction to machine learning. First, I am going to examine two sets of images. These images are provided from Yale, and the data sets contain "filtered" and "messy" data. The "filtered" data comes in the form of cropped pictures. Cropping them reduces the impact of the background on the SVD. The "messy" data comes in the form of uncropped pictures. These pictures still contain the background, and aren't as clean as the cropped. The goal is to examine the impact of the "filtered" data and the impact of the "noisy" data on the SVD and the accuracy in reconstruction.

Next, I am going to train 3 separate models to classify test data from a training set. The data for this set is going to be 5 second audio clips from songs. I will be testing three cases. These cases are: different musical genres and try to identify artist, Seattle based grunge bands and try to identify artist, and try to identify genre. After the data is created, I am going to train models using the Naive Bayes, Linear Discriminant Analysis (LDA), and K-nearest neighbors (KNN) algorithms. Then to cross validate, I will run the model 50 times to approximate the average accuracy of each method.

2 Theoretical Background

The details of SVD were discussed in a previous project. The short version is that it takes a matrix A and decomposes it into the matrices U, Σ, V. Σ is a diagonal matrix with the singular values on the main diagonal. These values indicate the principle components of our system. Using SVD, we can create a lower dimension representation of our system. This lower dimensional representation is useful for analyzing.

For the images in this project, I am going to examine the difference in the number of modes needed to recreate the original images from the cropped and uncropped pictures. Because the cropped image data set is larger than the uncropped image set, I am going to compare the percent of nodes required for both cropped and uncropped.

Moving on to the music, I am going to be doing something similar. to the faces. I am going to preform SVD on the data. But the data is going to the Fourier Transform of the music[1]. This gives me access to the frequencies occurring in these 5 second clips. This is important because the combination of frequencies are unique across genres and artists.[2]After preforming this SVD, I can then take the components of V that correspond to each sample, and use that to train the algorithms mentioned in the introduction. One thing to note is that all of these clustering algorithms are "supervised" algorithms. This is an advantage over "unsupervised" algorithms. Supervised algorithms implies that your training data has labels. Whereas unsupervised algorithms do not have labels on the training data.

2.1 Naive Bayes

The Naive Bayes classification algorithm assumes that you have n classes. Then, you can set up the following:

$$\frac{P(1|x)}{P(0|x)} = \frac{P(x|1)P(1)}{P(x|0)P(0)} \tag{1}$$

The RHS is very easy to calculate, because we know $P(1), P(0)$ and we can calculate $P(x|1), P(x|2)$. Matlab has a built in function: fitcnb(...). This function takes training data and labels and creates the Naive Bayes model.

2.2 Linear Discriminant Analysis

LDA and Naive Bayes share a lot of similarities. They are both built upon equation (1). However Naive Bayes assumes that the data is statistically independent. LDA does not assume this, but can only be applied to linear data sets, hence the name. LDA simply computes (1) and assigns the new data point to whichever category maximizes (1). LDA is the background of the classify(...) function within Matlab.

[1]It was suggested to preform a spectrogram of the data and SVD that. However because the length of the audio clip is 5 seconds, it is essentially a spectrogram with very large Δt. This kept the data sizes much more manageable than preforming the spectrogram

[2]Fourier Transform just takes the data and expands it into sines and cosines. This gives quick access to the frequency makeup of the data.

2.3 KNN

This is quite a simple algorithm. Because it is a supervised algorithm, it knows the group of all of the nearest neighbors. When it gets new data, it simply looks for the closest neighbor. Once it identifies the closest neighbor, it checks the group of that neighbor and classifies the new point as the same group as the nearest neighbor. Matlab's implementation of this algorithm is fitcknn(...).

3 Algorithm Implementation and Development

3.1 Part 1: Images

To develop this algorithm, I first needed to load the data. To do this, I made use of the dir function in Matlab, this function takes a string as input, and returns an object containing all of the items in that folder. Using lines 12-33, this loaded the first 13 files. The pictures provided skipped 14, so I had to split the loop to go from 1 to 13, and 15 to 39. While loading the data, I converted each image to grayscale. This makes it easier to manipulate the data.

The uncropped pictures were stuck in a .tar file, so I had to use the untar function in Matlab to extract. Then I used the same code for the cropped images to load the uncropped images. After loading the images and converting to grayscale, I reshaped into columns and stored them into my data matrix. The next step is to 0 the mean of the pictures. I accomplished this by subtracting out the "average" of the data set (Lines 86-97). Then I simply run SVD on the data/$\sqrt{n-1}$ as covered in the course notes. I then plotted the singular values, and ran the code to determine the appropriate value of r. I did this for both the cropped and uncropped. Then I used line 138 to calculate the percentage of singular nodes required to accurately recreate the images.

3.2 Part 2: Music Clustering

I first had to load all of the musical data. In order to do this, I used the dir function like in Part 1 to get all of the files in each folder. I had to delete useless files like album art. Once each song was loaded, I had to create 5 second snippets. That is achieved with lines 204-209. Each column represented a 5 second snippet. I repeated this process for each genre/artist I was testing. After that, I FFT'd the data to get the frequencies. The FFT'd data needed to be transposed in order to compute SVD. Lines 239 to 241 ensure that the ordering doesn't matter when constructing the data for SVD. I then plot the singular values to see how I can reduce the dimension of this system. I then plot the first 2 V projections for all 3 or 4 data sets. These are the 2 dominant directions, and they depict almost completely unique clusters. This is good because it helps the clustering algorithms described earlier attain better accuracy. Next, I construct a random training and testing set from my SVD'd data. The purpose of the randomness is to assure that order doesn't matter. Then, I simply trained the algorithms, and predicted with my test data. After obtaining the prediction, I compared with the expected solution. I ran this 50 times to ensure that order truly didn't matter, and that my results were accurate.

This technique is called cross-validation. I did the exact same thing for the rest of the cases. I just changed the data loaded in.

4 Computational Results

4.1 Part 1: Images

4.1.1 Cropped

In order to accurately recreate the data, r had to encompass 82.85% of all of the singular modes.

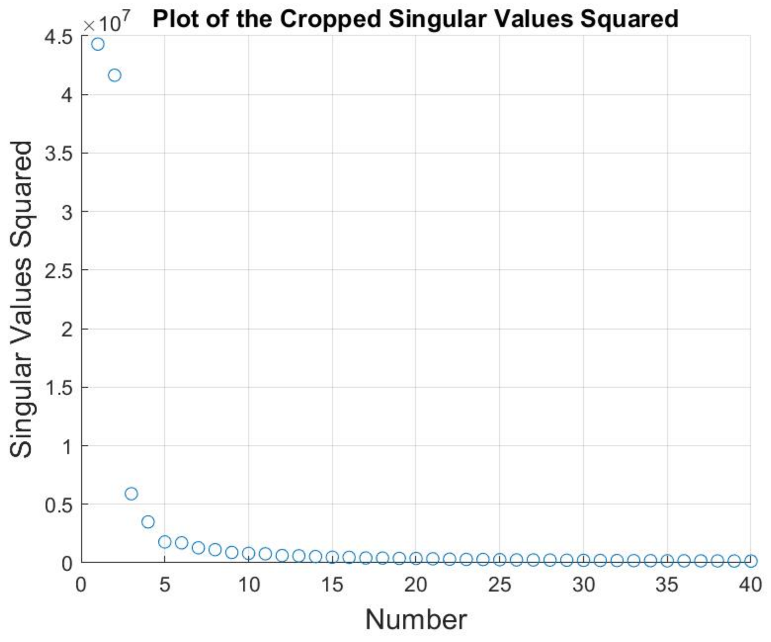

Figure 1: This shows the dominant modes for the cropped images.

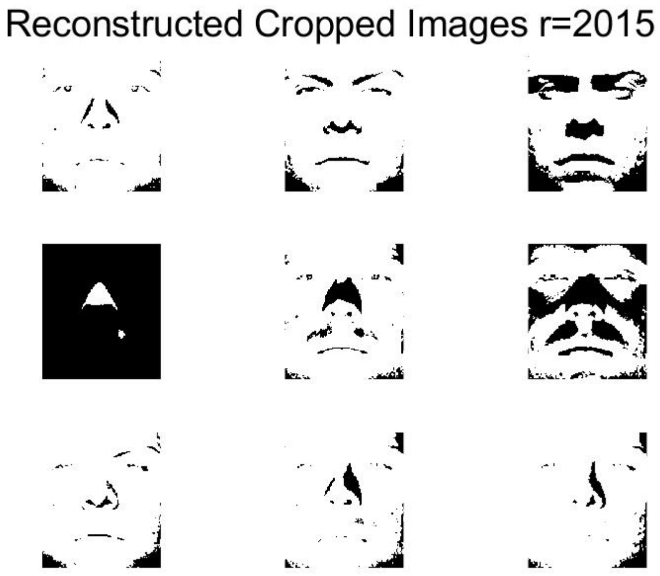

Figure 2: This shows the first 9 images reconstructed with = 118.

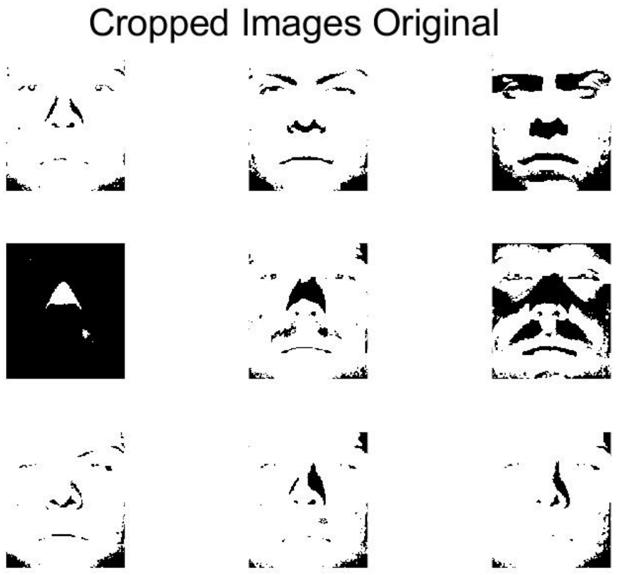

Figure 3: Compare figure 2 to figure 3, this verifies that this dimension reduction works.

4.1.2 Uncropped

In order to accurately recreate the data, r had to encompass 71.52% of all of the singular modes.

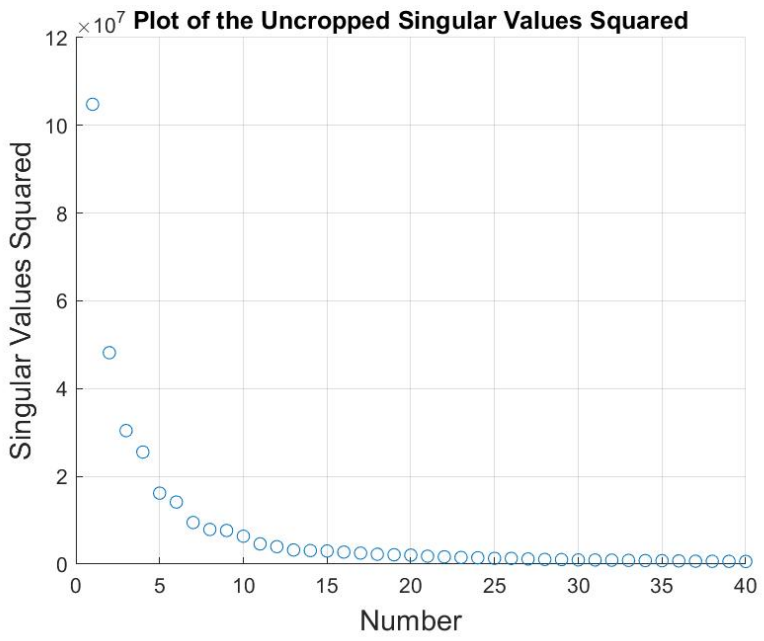

Figure 4: This shows the dominant modes for the uncropped images from Yale.

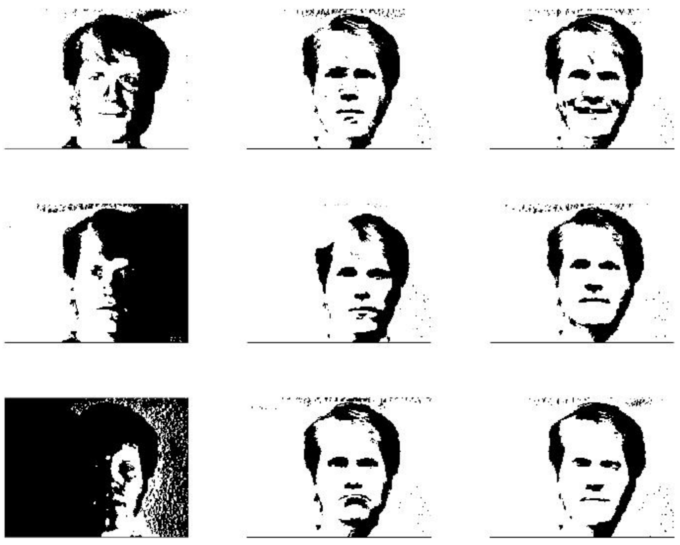

Figure 5: This shows the first 9 images reconstructed with $r = 118$.

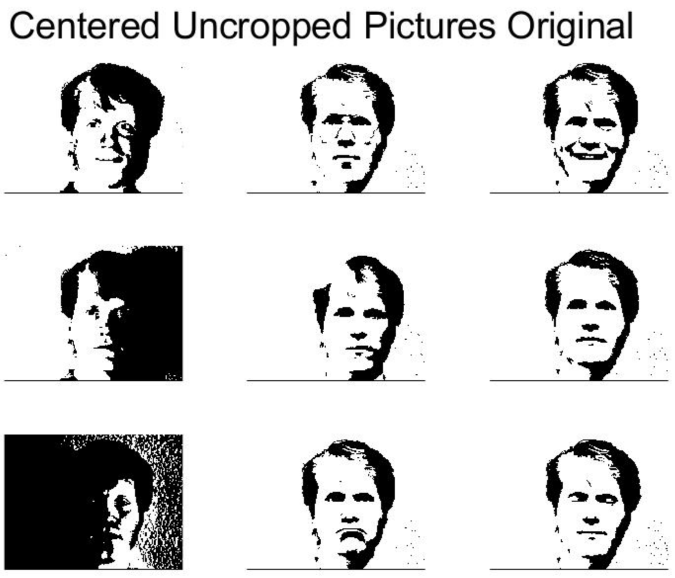

Figure 6: Compare figure 5 to figure 6 to verify that this dimension reduction works.

4.2 Part 2: Music Clustering

4.2.1 Case 1: Identify Artist of Separate Genres

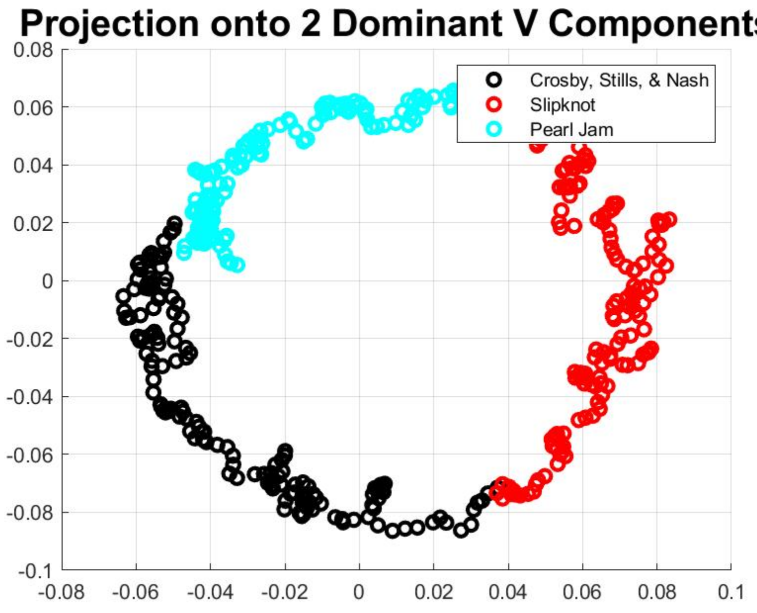

Figure: This projection shows that a clustering algorithm should be quite accurate in classifying artists of different genres

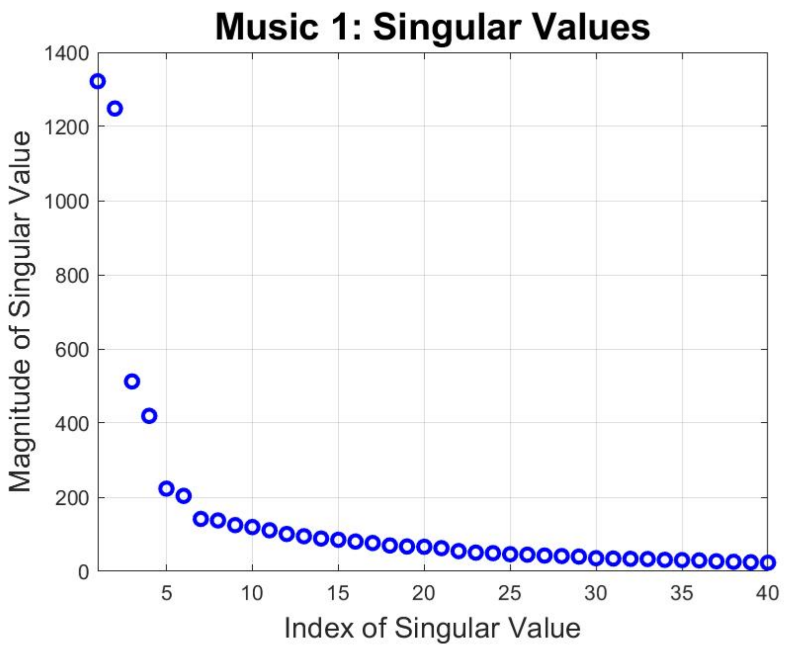

Figure: This shows the principle components for Case 1. We can reduce this matrix down to a 29 dimensional system.

4.2.2 Case 2: Identify Seattle Based Grunge Artists

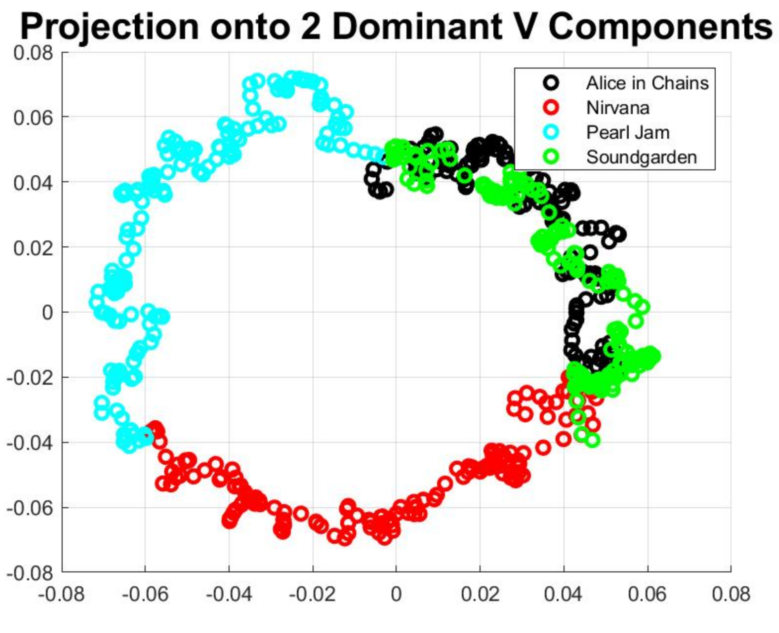

Figure: This projection shows that a clustering algorithm should be quite accurate in classifying Seattle based grunge bands. There may be an issue determining the difference between Alice in Chains and Soundgarden.

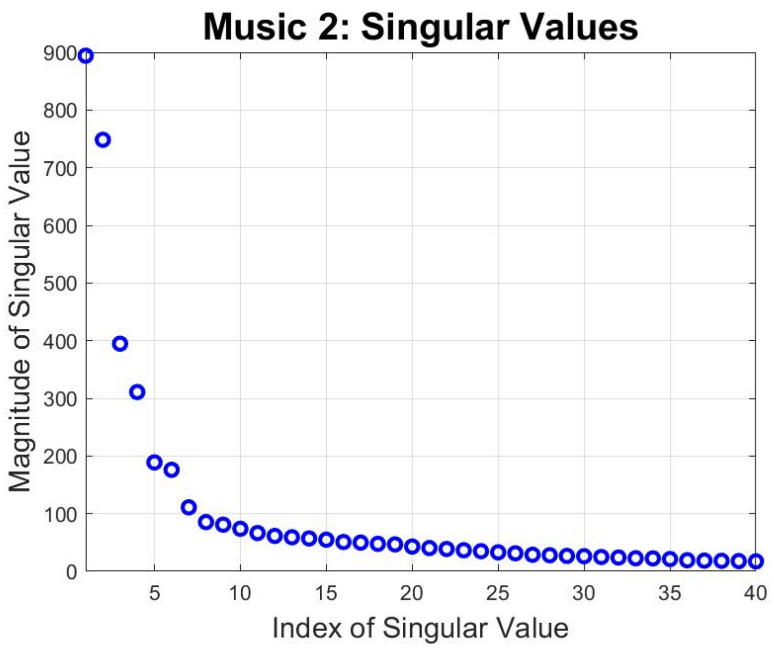

Figure: This shows the principle components for Case 1. We can reduce this matrix down to a 26 dimensional system.

4.2.3 Case 3: Identify Genre

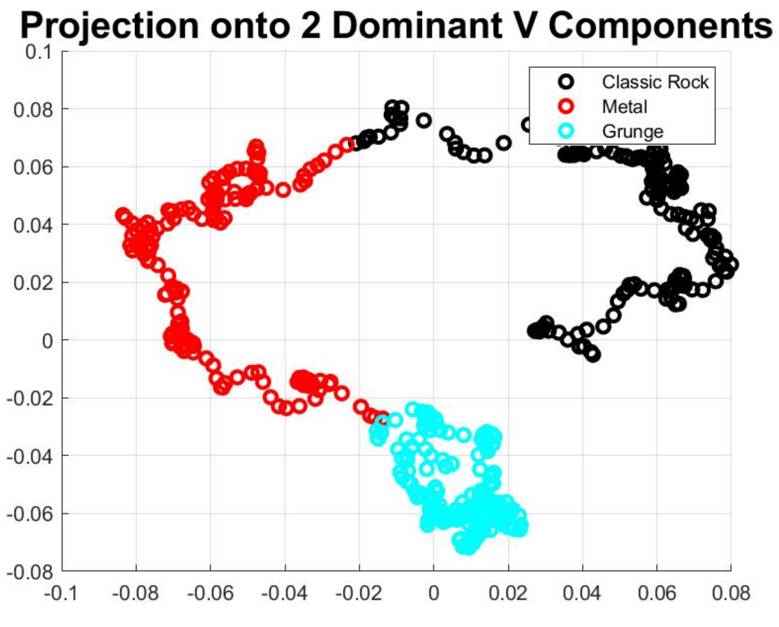

Figure: This projection shows that a clustering algorithm should be quite accurate in classifying different genres.

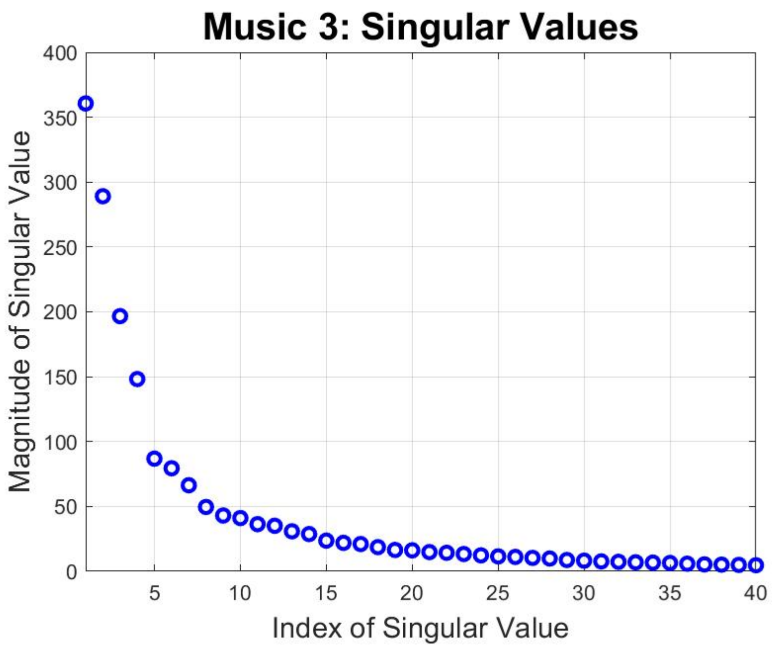

Figure: This shows the principle components for Case 1. We can reduce this matrix down to a 26 dimensional system.

5 Summary and Conclusions

5.1 Part 1: Images

From the figures above and the computational results, I can conclude that surprisingly, uncropped could be reduced more than the cropped photos. The way I rationalize this is due to the fact that the background across more of the photos is constant. This increases the covariance, and thus more of the energy is stored in earlier nodes. The interpretation of Σ is that it contains the value of the singular modes along its main diagonal. These values are the square root of the eigenvalues. Majorly non zero singular values indicate the the corresponding mode is important. U indicates the directions that actually matter, and V indicates how the image data projects onto the principle directions of U.

5.2 Part 2: Music Clustering

All three of the algorithms I implemented were extremely accurate. Using only 150 samples per artist or genre, I was able to achieve at least 98.57% accuracy. One of the important aspects of this homework I realized was randomly sampling all of the five second audio clips I made. In order to attain this randomness, I had to use lines like 239-242. This way I take a random sample of the clips I created. For Case 1, NB achieved an average accuracy of 98.87%, LDA achieved 99.33%, and KNN achieved 99.17%. For Case 2, NB achieved 99.05%, LDA achieved 99.43%, and KNN achieved 99.53%. And finally for Case 3, achieved 98.57%, LDA achieved 99.6%, and KNN achieved 99.67%. I cross validated these by taking 50 trials and randomizing the test data and the training data.

A MATLAB Functions and Implementation

Functions I used to implement the algorithm and why I used them:

zeros(...): This creates a matrix of zeros. Useful for pre-allocating the size that I want for my data.

[**'...',num2str(...)**]: This was useful for string concatenation. This allowed me to load the data efficiently.

dir(...): This loads all of the files within a folder.

rgb2gray(...): This converts the image from color to grayscale.

uint8(...): This converts whatever is within the parentheses into image data. Used to display images after mathematical manipulation.

[**U S V**]**=svd(...)**: Preforms the svd and returns the matrices U, S, and V. This was used to preform the PCA.

x,ylabel(...): The x or y label function sets the label of the corresponding axis to the specified label.

title(...): The title command sets the title of the active figure.

print(...): This print function saves the active figure to the name specified by the 1st input. The file type is determined by the 2nd input.

untar(...): This was used to extract the uncropped face data.

diags(...): This grabs the diagonal elements of the matrix. I used this to grab the value of the singular values and plot them. **double(...)**: This was used to convert the uint8 variables loaded by the load function to double variables. Casting to double means I can preform mathematical operations on them.

size(...): This was used to get the size of the 4th component of the data (number of frames). This is what determines how long the for loops in my code run.

reshape(...): This was used to take the 2-D data and convert it down to 1-D vector. It was important for tracking the paint can.

floor(...): Because I was tracking the can with pixel data, it doesn't make sense for a pixel location to provide a non-integer value, so this required that the location be an integer value.

imshow(...): This is used to display the each picture. This helped with visualization.

imread(...): This was used to read each of the pictures and store them in Matlab.

plot(...): Plot the provided data

scatter(...): Create a scatter plot from the given data.

fitcknn(...): Fit a k-nearest neighbor model to the data. Provide the training labels.

fitcnb(...): Fit a Naive Bayes model to the data. Provide the training labels.

classify(...): Preform LDA on the training data and the training labels. Provide test data to generate a prediction.

B MATLAB Code

```
%% Load Cropped Images
clear all, close all; clc
%Create variables
m=192;
n=168;
A=[];
avg=zeros(m*n,1);
count=0;
picPerPerson=64;

%Loop through first 13 files
for z=1:13
    s=['yaleB',num2str(z,'%02d')];
    list=dir(s);
    list=list(3:end);
    %load pictures
    for j=1:64
        f=list(j);
        u=imread([s,'\',f.name]);
        imshow(u)
        %Convert to grayscale if not already
        if(size(u,3)==1)
            M=double(u);
        else
            M=double(rgb2gray(u));
        end
    pause(0.1);
    R = reshape(M,size(M,1)*size(M,2),1);
    A = [A,R];
    avg = avg + R;
    count = count + 1;
    end
end
%% Load the second half of images
%Start at 15 because files skipped 14. do the same as above
for z=15:39
    s=['yaleB',num2str(z,'%02d')];
    list=dir(s);
    list=list(3:end);
    for j=1:64
        f=list(j);
        u=imread([s,'\',f.name]);
        imshow(u)
        if(size(u,3)==1)
            M=double(u);
        else
            M=double(rgb2gray(u));
        end
    pause(0.1);
    R = reshape(M,size(M,1)*size(M,2),1);
    A = [A,R];
    avg = avg + R;
    count = count + 1;
    end
end

%% Load Uncropped
s='yalefaces';
list=dir(s);
list=list(3:end);
%Create uncropped variables
m_uc=243;
n_uc=320;
A_uc=[];
avg_uc=zeros(m_uc*n_uc,1);
count_uc=0;
%loop through all the files and store them
for j=1:165
        f=list(j);
        u=imread([s,'\',f.name]);
        imshow(u)
        if(size(u,3)==1)
            M=double(u);
        else
            M=double(rgb2gray(u));
        end
    pause(0.1);
    R_uc = reshape(M,size(M,1)*size(M,2),1);
    A_uc = [A_uc,R_uc];
    avg_uc = avg_uc + R_uc;
    count_uc = count_uc + 1;
end


%% Calculate Average Face
avg=avg/count; %sum divided by the number
avgTS = uint8(reshape(avg,m,n));
figure(1), imshow(avgTS); %show the average face
%% Zero about the "Origin"
N=size(A,2);
%Subtract the average from each data point to give a mean of
   0
for j=1:N
    A(:,j) = A(:,j) - avg;
    R = reshape(A(:,j),m,n);
    imshow(R);
    pause(.1);
end

%% Compute the SVD, Visualize, and Recreate
clear VR V U S UR SR
%SVD
[U S V]=svd(A/sqrt(count-1),0);
%Examine the important nodes
figure(1)
scatter(1:2432,diag(S).^2)
xlim([0 40])
grid on
ylabel('Singular Values Squared','FontSize',14)
xlabel('Number','FontSize',14)
title('Plot of the Cropped Singular Values Squared','FontSize
   ',12)
print('C_SV','-djpeg')
r=2015;
%Dimension Reduction
UR(:,1:r)=U(:,1:r);
SR(1:r,1:r)=S(1:r,1:r);
V=V';
VR(1:r,:)=V(1:r,:);
%Recreate the original data
L=UR*SR*VR;
```

```
%%
figure(2)
subplot(2,1,1)
%Compare the original data
figure(2)
sgtitle('Reconstructed Cropped Images r=2015','FontSize',18)
for z=1:9
    subplot(3,3,z)
    imshow((uint8(25000*reshape(L(:,z),m,n))))
end
print('C_RC','-djpeg');
figure(3)
sgtitle('Cropped Images Original','FontSize',18)
for z=1:9
    subplot(3,3,z)
    imshow((uint8(25000*reshape(A(:,z),m,n))))
end
print('C_D','-djpeg');
recon_perc=(r/(length(S)))*100
%% Calculate Average Uncropped Face
%Do the same things for the cropped faces
avg_uc=avg_uc/count_uc;
aveUC=uint8(reshape(avg_uc,m_uc,n_uc));
figure(1),imshow(aveUC)
%% Center the Uncropped faces
N_uc=size(A_uc,2);
for j=1:N_uc
    A_uc(:,j)=A_uc(:,j)-avg_uc;
    R_uc=reshape(A_uc(:,j),m_uc,n_uc);
    imshow(R_uc);
    pause(.1);
end
%% Compute the Uncropped SVD, Visualize Modes, and
   Reconstruct
clear u_uc s_uc v_uc U_uc V_uc S_uc
[u_uc s_uc v_uc]=svd(A_uc/sqrt(N_uc-1),0);
scatter(1:165,diag(s_uc).^2)
xlim([0 40])
grid on
ylabel('Singular Values Squared','FontSize',14)
xlabel('Number','FontSize',14)
title('Plot of the Uncropped Singular Values Squared','
   FontSize',12)
print('UC_SV','-djpeg')
r_uc=118;
U_uc(:,1:r_uc)=u_uc(:,1:r_uc);
```

```
S_uc(1:r_uc,1:r_uc)=s_uc(1:r_uc,1:r_uc);
v_uc=v_uc';
V_uc(1:r_uc,:)=v_uc(1:r_uc,:);
L_uc=U_uc*S_uc*V_uc;
figure(2)
sgtitle('Uncropped Pictures Recreated r=118','FontSize',18)
for z=1:9
    subplot(3,3,z)
    imshow((uint8(25000*reshape(L_uc(:,z),m_uc,n_uc))))
end
print('UC_RC','-djpeg');
figure(3)
sgtitle('Centered Uncropped Pictures Original','FontSize',18)
for z=1:9
    subplot(3,3,z)
    imshow((uint8(25000*reshape(A_uc(:,z),m_uc,n_uc))))
end
print('UC_D','-djpeg');
ruc_perc=r_uc/length(diag(s_uc))*100
%% Music Classification Part 1: Band Classification
%% Load the Data
clear all;close all; clc
%Load Songs in directory
s='Classic Rock\Crosby, Stills & Nash\The Greatest Hits';
g='Metal\Antennas to Hell  The Best of Slipknot\';
l='Seattle_Grunge\Pearl Jam\Rearviewmirror (Pearl Jam
   Greatest Hits 1991-2003) Disc 1\';
%Create directory
list1=dir(s);
list2=dir(g);
list3=dir(l);
%Remove useless files
list1=list1(3:end-5);
list2=list2(3:end);
list3=list3(3:end-8);
P=[];
m=[];
%Load first 2 songs of Classic Rock
for z=1:2
    [x Fs]=audioread([s,'\',list1(z).name]);
    Xmono=(x(:,1)+x(:,2))/2;
    for j=1:floor(length(Xmono)/Fs)-5
        start=j*Fs;
        End=(j+5)*Fs;
        m=Xmono(start:End);
        P=[P,m];
    end
end
%Load first two songs of Metal
Metal=[];
for z=1:2
    [x Fs]=audioread([g,list2(z).name]);
    Xmono=(x(:,1)+x(:,2))/2;
    for j=1:floor(length(Xmono)/Fs)-5
        start=j*Fs;
        End=(j+5)*Fs;
        m=Xmono(start:End);
        Metal=[Metal,m];
    end
end
%Load first two songs of Grunge
Grunge=[];
    for z=1:2
       [x Fs]=audioread([l,'\',list3(z).name]);
       Xmono=(x(:,1)+x(:,2))/2;
        for j=1:floor(length(Xmono)/Fs)-5
            start=j*Fs;
            End=(j+5)*Fs;
            m=Xmono(start:End);
            Grunge=[Grunge,m];
        end
    end
%% FFT Data
clc, clear song_fft song_Spec
%Set a number of 5 second snippets to test
samp=150;
g1=randperm(samp);
g2=randperm(samp);
g3=randperm(samp);
X=[P(:,g1(1:samp)) Metal(:,1:g2(1:samp)) Grunge(:,1:g3(1:samp
   ))];

n = length(X);
t = (1:n)/Fs;
song_fft=[];
%FFT the data
for i = 1:3*samp
    song_fft=[song_fft;real(fftshift(fft(X(i,:))))];
end
%% Compute SVD
clear U S V KNN_AA KNN_OA NB_OA NB_AA DA_AA DA_OA
%FFT needs to be transposed
```

```
song_fft=song_fft';
[U S V]=svd(real(song_fft)/sqrt(3*samp-1),0);
figure(1)
plot(diag(S).^2,'bo','Linewidth',2)
xlim([1 40]);
grid on
xlabel('Index of Singular Value','FontSize',14)
ylabel('Magnitude of Singular Value','FontSize',14)
title('Music 1: Singular Values','FontSize',18)
print('MC1_Sing','-djpeg')
figure(2)
hold on
plot(V(1:samp,1),V(1:samp,2),'ko','Linewidth',2)
plot(V(samp+1:2*samp,1),V(samp+1:2*samp,2),'ro','Linewidth'
   ,2)
plot(V(2*samp+1:3*samp,1),V(2*samp+1:3*samp,2),'co','
   LineWidth',2)
hold off
grid on
title('Projection onto 2 Dominant V Components','FontSize'
   ,18)
legend('Crosby, Stills, & Nash','Slipknot','Pearl Jam')
print('MC1_Proj','-djpeg')
train=samp-20;
test=samp-train;
%% Train and Predict Model
clear q1 q2 q3 xClassic xMetal xGrunge xtrain xtest ctrain nb
    pre solution, close all
for z=1:50
    %Create random permutations
    q1=randperm(samp);
    q2=randperm(samp);
    q3=randperm(samp);
    %Create the actual solution (for comparing to prediction)
    solution=[ones(test,1);2*ones(test,1);3*ones(test,1)];

    %Create the training data
    xClassic=V(1:samp,1:29);
    xMetal=V(samp+1:2*samp,1:29);
    xGrunge=V(2*samp+1:3*samp,1:29 );

    %Put everything together into train and test
    xtrain=[xClassic(q1(1:train),:); xMetal(q2(1:train),:);
       xGrunge(q3(1:train),:)];
    xtest=[xClassic(q1(train+1:end),:); xMetal(q2(train+1:end
       ),:);xGrunge(q3(train+1:end),:)];
    ntest=size(xtest,1);
    % Naive Bayes
    ctrain=[ones(train,1); 2*ones(train,1);3*ones(train,1)];
    nb=fitcnb(xtrain,ctrain);
    % N-B Prediction
    pre=nb.predict(xtest);
    bar(pre)
    NB_acc=pre==solution;
    NB_OA(z)=(sum(NB_acc)/ntest)*100;
    % Discriminant Analysis
    SVM=classify(xtest,xtrain,ctrain);
    % DA Prediction
    bar(SVM)
    DA_acc=SVM==solution;
    DA_OA(z)=(sum(DA_acc)/ntest)*100;
    % K Nearest Neighbor
    idx=fitcknn(xtrain,ctrain);
    % KNN Predict
    pre_knn=idx.predict(xtest);
    bar(pre_knn)
    KNN_acc=pre_knn==solution;
    KNN_OA(z)=(sum(KNN_acc)/ntest)*100;
end
nave=size(NB_OA,2);
NB_AA=sum(NB_OA)/nave
DA_AA=sum(DA_OA)/nave
KNN_AA=sum(KNN_OA)/nave
%% Music Part 2: The Case for Seattle
%% Load the Data
clear all;close all; clc
s='Seattle_Grunge\Alice in Chains\Alice in Chains1\';
g='Seattle_Grunge\Nirvana\Nevermind\';
l='Seattle_Grunge\Pearl Jam\Rearviewmirror (Pearl Jam
   Greatest Hits 1991-2003) Disc 1';
d='Seattle_Grunge\Soundgarden\A-Sides\';
list1=dir(s);
list2=dir(g);
list3=dir(l);
list4=dir(d);
list1=list1(3:end);
list2=list2(3:end-5);
list3=list3(3:end-8);
list4=list4(3:end-6);
m=[];
%Load first 2 songs of Alice in Chains
AIC=[];
```

```
for z=1:2
    [x Fs]=audioread([s,list1(z).name]);
    Xmono=(x(:,1)+x(:,2))/2;
    for j=1:floor(length(Xmono)/Fs)-5
        start=j*Fs;
        End=(j+5)*Fs;
        m=Xmono(start:End);
        AIC=[AIC,m];
    end
end
%Load first two songs of Nirvana
Nirv=[];
for z=1:2
    [x Fs]=audioread([g,list2(z).name]);
    Xmono=(x(:,1)+x(:,2))/2;
    for j=1:floor(length(Xmono)/Fs)-5
        start=j*Fs;
        End=(j+5)*Fs;
        m=Xmono(start:End);
        Nirv=[Nirv,m];
    end
end
%Load first two songs of Pearl Jam
PJ=[];
    for z=1:2
       [x Fs]=audioread([l,'\',list3(z).name]);
       Xmono=(x(:,1)+x(:,2))/2;
        for j=1:floor(length(Xmono)/Fs)-5
            start=j*Fs;
            End=(j+5)*Fs;
            m=Xmono(start:End);
            PJ=[PJ,m];
        end
    end
%Load first two songs of Soundgarden
SG=[];
for z=1:2
       [x Fs]=audioread([d,list4(z).name]);
       Xmono=(x(:,1)+x(:,2))/2;
        for j=1:floor(length(Xmono)/Fs)-5
            start=j*Fs;
            End=(j+5)*Fs;
            m=Xmono(start:End);
            SG=[SG,m];
        end
    end
```

```
%% FFT Data
clc, clear song_fft song_Spec
%Set a number of 5 second snippets to test
samp=150;
g1=randperm(samp);
g2=randperm(samp);
g3=randperm(samp);
g4=randperm(samp);
X=[AIC(:,g1(1:samp)) Nirv(:,1:g2(1:samp)) PJ(:,1:g3(1:samp))
    SG(:,g4(1:samp))];
n = length(X);
t = (1:n)/Fs;
song_fft=[];
%FFT the data
for i = 1:4*samp
    song_fft=[song_fft;real(fftshift(fft(X(i,:))))];
end
%% Compute SVD
clear U S V KNN_AA KNN_OA NB_OA NB_AA DA_AA DA_OA
%FFT needs to be transposed
song_fft=song_fft';
[U S V]=svd(real(song_fft)/sqrt(4*samp-1),0);
figure(1)
plot(diag(S).^2,'bo','Linewidth',2)
xlim([1 40]);
grid on
xlabel('Index of Singular Value','FontSize',14)
ylabel('Magnitude of Singular Value','FontSize',14)
title('Music 2: Singular Values','FontSize',18)
print('MC2_Sing','-djpeg')
figure(2)
hold on
plot(V(1:samp,1),V(1:samp,2),'ko','Linewidth',2)
plot(V(samp+1:2*samp,1),V(samp+1:2*samp,2),'ro','Linewidth'
    ,2)
plot(V(2*samp+1:3*samp,1),V(2*samp+1:3*samp,2),'co','
    LineWidth',2)
plot(V(3*samp+1:4*samp,1),V(3*samp+1:4*samp,2),'go','
    LineWidth',2)
hold off
grid on
title('Projection onto 2 Dominant V Components','FontSize'
    ,18)
legend('Alice in Chains','Nirvana','Pearl Jam','Soundgarden')
print('MC2_Proj','-djpeg')
train=samp-20;
```

```
test=samp-train;
%% Train and Predict Model
clear q1 q2 q3 xClassic xMetal xGrunge xtrain xtest ctrain nb
     pre solution, close all
for z=1:50
    %Create random permutations
    q1=randperm(samp);
    q2=randperm(samp);
    q3=randperm(samp);
    q4=randperm(samp);
    %Create the actual solution (for comparing to prediction)
    solution=[ones(test,1);2*ones(test,1);3*ones(test,1);4*
       ones(test,1)];

    %Create the training data
    xAIC=V(1:samp,1:26);
    xNirv=V(samp+1:2*samp,1:26);
    xPJ=V(2*samp+1:3*samp,1:26);
    xSG=V(3*samp+1:4*samp,1:26);

    %Put everything together into train and test
    xtrain=[xAIC(q1(1:train),:); xNirv(q2(1:train),:);xPJ(q3
       (1:train),:);xSG(q4(1:train),:)];
    xtest=[xAIC(q1(train+1:end),:); xNirv(q2(train+1:end),:);
       xPJ(q3(train+1:end),:);xSG(q4(train+1:end),:)];
    ntest=size(xtest,1);
    % Naive Bayes
    ctrain=[ones(train,1); 2*ones(train,1);3*ones(train,1);4*
       ones(train,1)];
    nb=fitcnb(xtrain,ctrain);
    % N-B Prediction
    pre=nb.predict(xtest);
    bar(pre)
    NB_acc=pre==solution;
    NB_OA(z)=(sum(NB_acc)/ntest)*100;
    % Discriminant Analysis
    SVM=classify(xtest,xtrain,ctrain);
    % DA Prediction
    bar(SVM)
    DA_acc=SVM==solution;
    DA_OA(z)=(sum(DA_acc)/ntest)*100;
    % K Nearest Neighbor
    idx=fitcknn(xtrain,ctrain);
    % KNN Predict
    pre_knn=idx.predict(xtest);
    bar(pre_knn)
    KNN_acc=pre_knn==solution;
    KNN_OA(z)=(sum(KNN_acc)/ntest)*100;
end
nave=size(NB_OA,2);
NB_AA=sum(NB_OA)/nave
DA_AA=sum(DA_OA)/nave
KNN_AA=sum(KNN_OA)/nave

%% Music Classification Part 3: Genre Classification
%% Load the Data
clear all;close all; clc
%Load Songs in directory
CR1='Classic Rock\Crosby, Stills & Nash\The Greatest Hits\05
   Marrakesh Express.mp3';
CR2='Classic Rock\Steve Miller Band\Wide River\06 Horse and
   Rider.wma';
M1='Metal\Antennas to Hell  The Best of Slipknot\16 Sulfur.
   wma';
M2='Metal\Somewhere Back in Time  The Best of 1980-1989\04
   The Trooper.wma';
G1='Seattle_Grunge\Pearl Jam\Rearviewmirror (Pearl Jam
   Greatest Hits 1991-2003) Disc 1\04 Jeremy.mp3';
G2='Seattle_Grunge\Alice in Chains\Alice In Chains2\28 Would
   (disc 2).mp3';

CR=[];
m=[];
%Load first 2 songs of Classic Rock
[x Fs]=audioread(CR1);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    CR=[CR,m];
end
[x Fs]=audioread(CR2);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    CR=[CR,m];
end
%Load first two songs of Metal
Metal=[];
```

```matlab
[x Fs]=audioread(M1);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    Metal=[Metal,m];
end
[x Fs]=audioread(M2);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    Metal=[Metal,m];
end
%Load first two songs of Grunge
Grunge=[];
[x Fs]=audioread(G1);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    Grunge=[Grunge,m];
end
[x Fs]=audioread(G2);
Xmono=(x(:,1)+x(:,2))/2;
for j=1:floor(length(Xmono)/Fs)-5
    start=j*Fs;
    End=(j+5)*Fs;
    m=Xmono(start:End);
    Grunge=[Grunge,m];
end
%% FFT Data
clc, clear song_fft song_Spec
%Set a number of 5 second snippets to test
samp=150;
g1=randperm(samp);
g2=randperm(samp);
g3=randperm(samp);
X=[CR(:,g1(1:samp)) Metal(:,1:g2(1:samp)) Grunge(:,1:g3(1:
    samp))];
n = length(X);
t = (1:n)/Fs;
song_fft=[];
```

```
%FFT the data
for i = 1:3*samp
    song_fft=[song_fft;real(fftshift(fft(X(i,:))))];
end
%% Compute SVD
clear U S V KNN_AA KNN_OA NB_OA NB_AA DA_AA DA_OA
%FFT needs to be transposed
song_fft=song_fft';
[U S V]=svd(real(song_fft)/sqrt(3*samp-1),0);
figure(1)
plot(diag(S).^2,'bo','Linewidth',2)
xlim([1 40]);
grid on
xlabel('Index of Singular Value','FontSize',14)
ylabel('Magnitude of Singular Value','FontSize',14)
title('Music 3: Singular Values','FontSize',18)
print('MC3_Sing','-djpeg')
figure(2)
hold on
plot(V(1:samp,1),V(1:samp,2),'ko','Linewidth',2)
plot(V(samp+1:2*samp,1),V(samp+1:2*samp,2),'ro','Linewidth'
   ,2)
plot(V(2*samp+1:3*samp,1),V(2*samp+1:3*samp,2),'co','
   LineWidth',2)
hold off
grid on
title('Projection onto 2 Dominant V Components','FontSize'
   ,18)
legend('Classic Rock','Metal','Grunge')
print('MC3_Proj','-djpeg')
train=samp-20;
test=samp-train;
%% Train and Predict Model
clear q1 q2 q3 xClassic xMetal xGrunge xtrain xtest ctrain nb
    pre solution, close all
for z=1:50
    %Create random permutations
    q1=randperm(samp);
    q2=randperm(samp);
    q3=randperm(samp);
    %Create the actual solution (for comparing to prediction)
    solution=[ones(test,1);2*ones(test,1);3*ones(test,1)];

    %Create the training data
    xClassic=V(1:samp,1:26);
    xMetal=V(samp+1:2*samp,1:26);
    xGrunge=V(2*samp+1:3*samp,1:26);

    %Put everything together into train and test
    xtrain=[xClassic(q1(1:train),:); xMetal(q2(1:train),:);
       xGrunge(q3(1:train),:)];
    xtest=[xClassic(q1(train+1:end),:); xMetal(q2(train+1:end
       ),:);xGrunge(q3(train+1:end),:)];
    ntest=size(xtest,1);
    % Naive Bayes
    ctrain=[ones(train,1); 2*ones(train,1);3*ones(train,1)];
    nb=fitcnb(xtrain,ctrain);
    % N-B Prediction
    pre=nb.predict(xtest);
    bar(pre)
    NB_acc=pre==solution;
    NB_OA(z)=(sum(NB_acc)/ntest)*100;
    % Discriminant Analysis
    SVM=classify(xtest,xtrain,ctrain);
    % DA Prediction
    bar(SVM)
    DA_acc=SVM==solution;
    DA_OA(z)=(sum(DA_acc)/ntest)*100;
    % K Nearest Neighbor
    idx=fitcknn(xtrain,ctrain);
    % KNN Predict
    pre_knn=idx.predict(xtest);
    bar(pre_knn)
    KNN_acc=pre_knn==solution;
    KNN_OA(z)=(sum(KNN_acc)/ntest)*100;
end
nave=size(NB_OA,2);
NB_AA=sum(NB_OA)/nave
DA_AA=sum(DA_OA)/nave
KNN_AA=sum(KNN_OA)/nave
```