AMATH 582 Homework 4

Shannon Dow

March, 2020

Abstract

This paper will demonstrate the usefulness and power of the Singular Value Decomposition (SVD) as a computational tool, first in creating low rank approximations of and analyzing faces, and then in classification. The first part of this assignment uses the SVD to compare low rank approximations and dominant modes of data sets of cropped and un-cropped faces. The second part uses the SVD as a part of training a program to recognize song data. To classify the music, Linear Discriminant Analysis (LDA) tools will be used to find an optimal subspace to separate the data classes.

1 Introduction and Overview

1.1 Part 1: Yale Faces

For this part, there are two datasets: a large one with many cropped pictures of people with a serious expression, and a smaller one of people's entire head with different expressions. In performing a Singular Value Decomposition (SVD) on both sets of images, I will determine the dominant modes needed to reconstruct the data, and form low rank approximations to the images.

1.2 Part 2: Music Classification

The goal of this section is to train a computer program to distinguish between different types of music.
Test 1 I will build a training set using 5 second clips of Mozart (Classical), The Eagles (Rock), and Snoop Dogg (Rap) and see how well the computer can identify other clips from the same artists
Test 2 For this test, I will repeat but build a training set with 3 bands (The Eagles, Fleetwood Mac, and Journey) from the same genre (Rock) and see if the program can identify bands.
Test 3 For this, I will use multiple bands from the same three genres as test 1 and evaluate the computer's ability to identify genre. To achieve these results, I will be using higher level MATLAB implementation to perform Linear Discriminant Analysis.

2 Theoretical Background

2.1 Part 1: Yale Faces

For this section, we are using the SVD to identify correlations between large data sets of people's faces. To create the A matrix to perform the SVD, I first reshape the image data into columns of numbers and add them into a matrix. The reason SVD is such a powerful computational tool for this overdetermined system is because all matrices have a SVD. Once we have the reduced SVD, it can be interpreted in the following way:

- **U** The columns of U represent the modes, or a set of characteristics of the face. The first column of U, corresponding to the largest singular value, is the most dominant feature. These are also called eigenfaces.
- Σ This is the singular values. The number of relevant (comparatively large) singular values tells you how many modes it takes to represent the images

- **V** The V values give the coefficients of the eigenfaces. They tell you how much of each eigenface you need to get the best approximation.

The main reason for using the SVD here is to look at low rank approximations as well as relevant modes. To obtain low rank approximations:

$$Rank \text{ n } approximation = \sum_{j=1}^{n} V_j * \Sigma_{jj} * U_j$$

2.2 Part 2: Music Classification

In order to perform supervised machine learning, the first goal is to take the data and make a training set, where the labels of the different categories are known. LDA finds a subspace for optimal separation of the data by identifying the projection that maximizes the distance between the different categories. Then, for linear analysis, it will create a line that provides the best separation of the different classes. If a quadratic analysis, it would choose the best quadratic to separate the data.

For this project, since we are working with 5 second segments of song data, we want to have the frequency and time information. For this we will need to find the spectrogram. To get the spectrogram, we will use a Gaussian Gabor transform, or a short time Fourier transform. This slides a filter along the music signal in time and gets localized data there about the frequency.

Once we have the spectrogram data from each song snippet, before we perform LDA, we want to take the reduced Singular Value Decomposition (SVD). To decrease computational time, we only want to keep the relevant modes, or columns of V, so it is helpful to look at the dominant, or largest singular values corresponding to those columns. Then we can use this truncated V matrix as the training data for our LDA. I will also use a similarly shaped truncated V from the test data when performing LDA.

3 Algorithm Implementation and Development

3.1 Part 1: Yale Faces

Before performing the analysis on the data using the SVD, it is first key to have the data in a usable format. The first section of the code takes the image files out of the folders of data. Then, each image is converted to a double data type and reshaped into a column vector. The result is then added to the matrix A.
The dimensions of A are: (size of image(hight x width) x number of images) .
The SVD is then obtained using MATLAB's reduced SVD function. To analyze the modes, you must reshape the columns of U back into images. Since U is a unitary matrix, the columns of U are orthonormal, so they are scaled to have the norm 1. In order to view the image, these columns need to be rescaled back into 0-255 scale. This is done by subtracting off the minimum and multiplying by $\frac{255}{max}$ or each column.

3.2 Part 2: Music Classification

Again, the first part of this assignment is to get the data into a usable format. To do this, we must first get .wav files of the music samples we want to use. Then as the .wav files are very large, it is useful to re-sample the music data so that we decrease computational time. The goal is to form a matrix where each column is a vectorized spectrogram of a 5-second snippet of a song, one for the training data, and one for the test data, created the same way, (See Algorithm 1).

Once we have the two matrices of test and training data, we take the reduced SVD. By analyzing the number of relevant singular values, we know how many columns of V we need. Next, using the MATLAB function `classify`, we can perform LDA. `classify` takes in 3 main arguments, the test and training data, a vector containing the labels for the training data.

Once we have the computer's classifications of the test data, we can compare them to actual known labels and see how accurate the program is.

Algorithm 1: Updating the Matrix of Vectorized Song Data

Let count be the number of the column you want to start updating A on
for The number of desired additional columns **do**
  Read in a 5 second sample from the data
  Re-sample the 5 second sample
  Find the spectrogram using the Gabor Window
  Reshape the spectrogram into a vector
  Add that vector to A
  Update count
end for

Figure 1 Singular Values (Cropped)

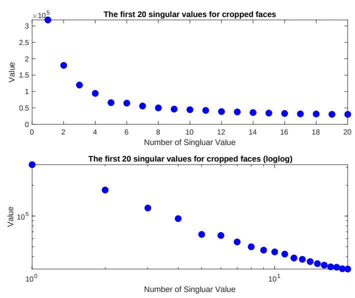
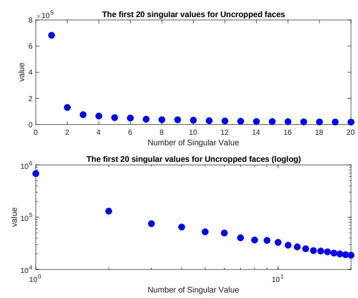


Figure 2 Singular Values (Uncropped)



4 Computational Results

4.1 Part 1: Yale Faces

Figures 1 through 10 show the main differences between the SVD analysis on the cropped and un-cropped Yale faces. Comparing Figures 1 and 2, there are more large singular values for the cropped images. This is because it is easier for the SVD to pick up correlations in the images when everything is in roughly the same place. This can also be seen in later figures, that low rank approximations are more descriptive for the cropped images, and you can clearly different sets of features in the modes.

Figure 3 First 3 Modes All Faces (Cropped)

Figure 4 First 3 Modes All Faces (Uncropped)

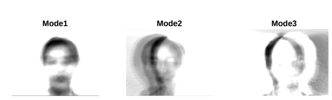


Figure 5 First 3 Modes One Face (Cropped)

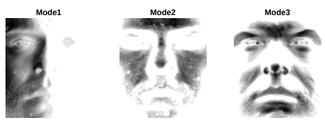


Figure 6 First 3 Modes One Face (Uncropped)

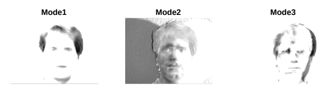


Figure 7 Rank Approximations One Face (Cropped)

Figure 8 Rank Approximations One Face (Uncropped)

Figure 9 Rank Approximations All Faces (Cropped)

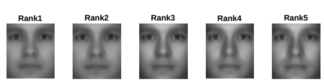
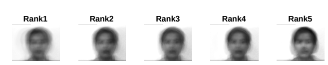


Figure 10 Rank Approximations All Faces (Uncropped)



4.2 Part 2: Music Classification

4.2.1 Test 1: 3 Different Bands from 3 different genres

For this test, I used songs from Mozart (Classical), The Eagles(Rock), and Snoop Dogg(Rap) to create a training set. Then used other song data from the same bands to create a test set. This test has 81.6%accuracy.

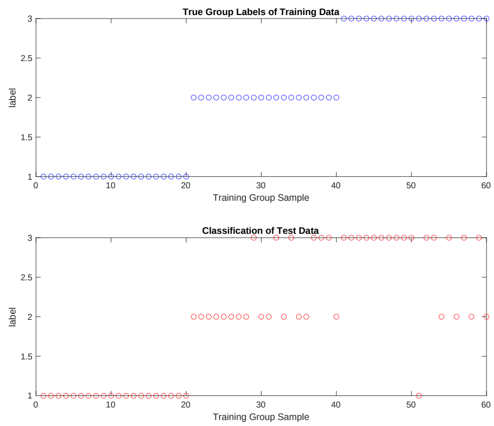


Figure 1: Mozart(1), The Eagles(2), and Snoop Dogg(3)

4.2.2 Test 2: 3 different Bands from the same genre

For this test, I used The Eagles, Journey, and Fleetwood Mac as soft rock bands rom the 60s/70s. This test has 65%accuracy.

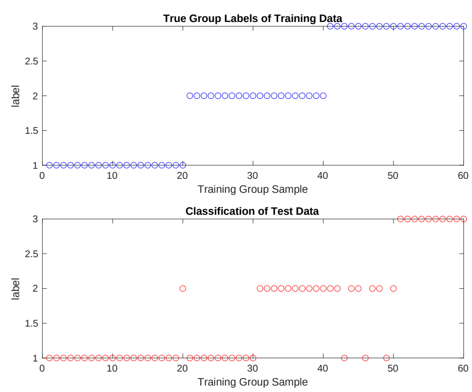


Figure 2: The Eagles(1), Journey(2), and Fleetwood Mac(3)

4.2.3 Test 3: Multiple Bands from three different genres

For this test, using the same three genres as above, I chose multiple artists from each.

- Classical Mozart, Bach, Beethoven, and Chopin
- Rock The Eagles, Journey, and Fleetwoord Mac
- Rap Snoop Dogg, Eminem, 50 Cent

This test has has 88.33% accuracy.

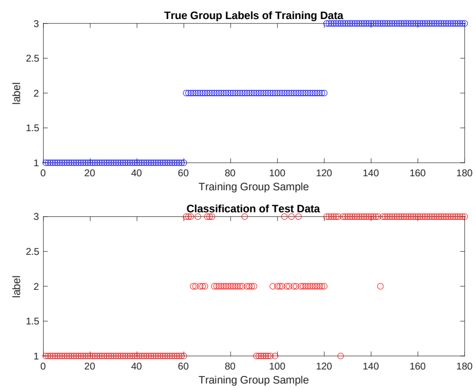


Figure 3: Classical(1), Rock(2), and Rap(3)

5 Summary and Conclusions

5.1 Part 1: Yale Faces

By analyzing the results from the SVD analysis, you can see that the SVD works better when the data is set up in a convenient way. For this, when all the faces are aligned and making the same facial expression, it is easier to find dominant shared characteristics. It also makes our low-rank approximations more accurate.

5.2 Part 2: Music Classification

From these classification tests, the most accurate test was the final one that distinguished between genres. One explanation for this is that I used a larger training and test data set. Since it had more, different information, it was better able to classify the test data. Additionally, I did choose relatively different sounding genres. The most often mis-labeled was the soft rock which is in the middle of classical and rap. As expected, it was the most difficult for the program to distinguish between artists in the same genre.

Appendix A MATLAB Functions

- `dir` Gets the directory

- `imread` Takes data and turns it into the uint8 image datatype
- `double` Converts to the double data type
- `reshape(a,x,y)` The reshape function takes data (a) and turns it into a matrix of the size (x by y).
- `imshow` This displays an image of type uint8
- `classify(Test, Train, Label)` Classify performs discriminant analysis on data once it receives training data with matching labels. It then classifies the test data into the categories listed in Label based on the training data.

Appendix B MATLAB Code

B.1 GITHUB LINK

https://github.com/shannondow/AMATH-582-Homeworks-2020

```
%% Cropped Faces:

D = '/Users/shannondow/MATLAB/projects/582HW4/yalefaces_cropped/CroppedYale';
S = dir(fullfile(D,'*'));
N = setdiff({S([S.isdir]).name},{}); % list of subfolders of D
A = [];
count = 1;
for ii = 1:numel(N)
    T = dir(fullfile(D,N{ii},'*.pgm')); % improve by specifying the file
        extension.
    C = {T(~[T.isdir]).name}; % files in subfolder.
    for jj = 1:numel(C)
        F = fullfile(D,N{ii},C{jj});
        a = imread((T(jj).name));
        aa = double(a);
        [m,n]=size(aa);
        x=reshape(aa,m*n,1);
        x = x-mean(x);
        A(:,count) = x;
        count = count+1;
    end
end
%Take The SVD
[U,Sig,V] = svd(A,0);
% Plot the Singular Values:

figure(1)
for i = 1:20
    subplot(2,1,1)
    plot(i,Sig(i,i),'b.','MarkerSize',30)
    hold on;
    title('The first 20 singular values for cropped faces')
    xlabel('Number of Singluar Value');
    ylabel('Value')
    subplot(2,1,2)
    loglog(i,Sig(i,i),'b.','MarkerSize',30)
    hold on;
    title('The first 20 singular values for cropped faces (loglog)')
```

```
    xlabel('Number of Singluar Value');
    ylabel('Value')
end

% Low Rank Approximations:
figure(2)
for rank = 1:5
    Approx = U(:,1:rank)*Sig(1:rank,1:rank)*V(:,1:rank)';
    App = reshape(Approx(:,1), m,n);
    subplot(1,5,rank)
    imshow(uint8(App))
    nam = strcat('Rank',num2str(rank));
    title(nam);
end
% Modes (Columns of U)
% Rescale to 255
for j = 1:length(U(1,:))
    minval = min(U(:,j));
    maxval = max(U(:,j));
    U(:,j) = U(:,j) - minval;
    U(:,j) = U(:,j)*(255/maxval);
end
figure(3)
for mode = 1:3
    yy= reshape(U(:,mode), m,n);
    subplot(1,3,mode)
    imshow(uint8(yy))
    nam = strcat('Mode',num2str(mode));
    title(nam);
end

%%
%% Just Subject 1:
[U1,Sig1,V1] = svd(A(:,1:64),0);

%Plot Singular Values
figure(4)
for i = 1:20
    subplot(2,1,1)
    plot(i,Sig1(i,i),'b.','MarkerSize',30)
    hold on;
    title('Singular Values Subject 1 Uncropped')
    xlabel('Number of Singluar Value');
    ylabel('Value')
    subplot(2,1,2)
    loglog(i,Sig1(i,i),'b.','MarkerSize',30)
    hold on;
    title('Singular Values Subject 1 Uncropped (loglog)')
    xlabel('Number of Singluar Value');
    ylabel('Value')
end
 %Low Rank Approximations:
figure(5)
for rank = 1:5
```

```
    Approx = U1(:,1:rank)*Sig1(1:rank,1:rank)*V1(:,1:rank)';
    App = reshape(Approx(:,1), m,n);
    subplot(1,5,rank)
    imshow(uint8(App))
    nam = strcat('Rank',num2str(rank));
    title(nam);
end

% Modes (Columns of U)
for j = 1:length(U1(1,:))
    minval = min(U1(:,j));
    maxval = max(U1(:,j));
    U1(:,j) = U1(:,j) - minval;
    U1(:,j) = U1(:,j)*(255/maxval);
end
figure(6)
for mode = 1:3
    yy= reshape(U1(:,mode), m,n);
    subplot(1,3,mode)
    imshow(uint8(yy))
    nam = strcat('Mode',num2str(mode));
    title(nam);
end
%% Uncropped Faces
%untar('yalefaces_uncropped.tar')
D = '/Users/shannondow/MATLAB/projects/582HW4/yalefaces';
S = dir(fullfile(D,'*'));
A = [];
count = 1;
for ii = 3:numel(S)
    a = imread(S(ii).name);
    aa = double(a);
    [m,n]=size(aa);
    x=reshape(aa,m*n,1);
    %x = x-mean(x);
    A(:,count) = x;
    count = count+ 1;
end

%Take The SVD
[U,Sig,V] = svd(A,0);
% Plot the Singular Values:

figure(1)
for i = 1:20
    subplot(2,1,1)
    plot(i,Sig(i,i),'b.','MarkerSize',30)
    hold on;
    title('The first 20 singular values for Uncropped faces')
    xlabel('Number of Singular Value')
    ylabel('value')
    subplot(2,1,2)
    loglog(i,Sig(i,i),'b.','MarkerSize',30)
    hold on;
```

```
    title('The first 20 singular values for Uncropped faces (loglog)')
    xlabel('Number of Singular Value')
    ylabel('value')
end

% Low Rank Approximations:
figure(2)
for rank = 1:5
    Approx = U(:,1:rank)*Sig(1:rank,1:rank)*V(:,1:rank)';
    App = reshape(Approx(:,1), m,n);
    subplot(1,5,rank)
    imshow(uint8(App))
    nam = strcat('Rank',num2str(rank));
    title(nam);
end
% Modes (Columns of U)
for j = 1:length(U(1,:))
    minval = min(U(:,j));
    maxval = max(U(:,j));
    U(:,j) = U(:,j) - minval;
    U(:,j) = U(:,j)*(255/maxval);
end
figure(3)
for mode = 1:3
    yy= reshape(U(:,mode), m,n);
    subplot(1,3,mode)
    imshow(uint8(yy))
    nam = strcat('Mode',num2str(mode));
    title(nam);
end

%% Just Subject 1:
[U1,Sig1,V1] = svd(A(:,1:11),0);

%Plot Singular Values
figure(4)
for i = 1:11
    subplot(2,1,1)
    plot(i,Sig1(i,i),'b.','MarkerSize',30)
    hold on;
    title('Singular Values Subject 1 Uncropped')
    xlabel('Number of Singular Value')
    ylabel('value')
    subplot(2,1,2)
    loglog(i,Sig1(i,i),'b.','MarkerSize',30)
    hold on;
    title('Singular Values Subject 1 Uncropped (loglog)')
    xlabel('Number of Singular Value')
    ylabel('value')
end
 %Low Rank Approximations:
figure(5)
for rank = 1:5
    Approx = U1(:,1:rank)*Sig1(1:rank,1:rank)*V1(:,1:rank)';
```

```
    App = reshape(Approx(:,1), m,n);
    subplot(1,5,rank)
    imshow(uint8(App))
    nam = strcat('Rank',num2str(rank));
    title(nam);
end

% Modes (Columns of U)
for j = 1:length(U1(1,:))
    minval = min(U1(:,j));
    maxval = max(U1(:,j));
    U1(:,j) = U1(:,j) - minval;
    U1(:,j) = U1(:,j)*(255/maxval);
end
figure(6)
for mode = 1:3
    yy= reshape(U1(:,mode), m,n);
    subplot(1,3,mode)
    imshow(uint8(yy))
    nam = strcat('Mode',num2str(mode));
    title(nam);
end
%% Part 2: Test 1-
%Create Training Set
A = [];
count = 1;
[count, A] = updateA('classical.wav',count,A,30,20);
[count, A] = updateA('EaglesS1.wav',count,A,30,10);
[count, A] = updateA('EaglesS2.wav',count,A,30,10);
[count, A] = updateA('Snoop1.wav',count,A,30,10);
[count, A] = updateA('Snoop2.wav',count,A,30,10);
%A = A'

[UA,SA,VA] = svd(A,0);
VA = VA(:,1:10);
%Create Test Data
B = [];
count = 1;
[count, B] = updateA('classical.wav', count, B, 200,20);
[count, B] = updateA('EaglesS3.wav', count, B, 30,10);
[count, B] = updateA('EaglesS4.wav', count, B, 30,10);
[count, B] = updateA('Snoop2.wav', count, B, 200,10);
[count, B] = updateA('Snoop3.wav', count, B, 30,10);
%B = B';

[UB,SB,VB] = svd(B,0);
VB = VB(:,1:10);
% Make the group names
group = [];
for i = 1:20
    group(i) = 1;
end
for i = 21:40
    group(i) = 2;
```

```
end
for i = 41:60
    group(i) = 3;
end
group = group';

%%
[class,err,POSTERIOR,logp,coeff] = classify(VB,VA,group,'linear');

VA = VA';
classic = [];
eagles = [];
snoop = [];
for i = 1:60
    if (0<i)&& (i<21)
        classic = [classic;norm(VA(:,i))];
    elseif (21 <= i)&&(i < 41)
        eagles = [eagles;norm(VA(:,i))];
    elseif (41 <= i) && (i<61)
        snoop = [snoop;norm(VA(:,i))];
    end
end
all = [classic;eagles;snoop];

%%
figure(4)
subplot(2,1,1)
plot(group,'bo')
title('True Group Labels of Training Data')
xlabel('Training Group Sample')
ylabel('label')
subplot(2,1,2)
plot(class,'ro')
title('Classification of Test Data')
xlabel('Training Group Sample')
ylabel('label')

%%
%calculate error
corr = 0;
for i = 1:length(group)
    if group(i) == class(i)
        corr =corr+ 1;
    end
end
error = (length(group)-corr)/(length(group))*100;
disp(error);

%%
figure(5)
plot(classic,'bo')
hold on
plot(eagles,'ro')
hold on
```

```
plot(snoop,'go')
hold on
%% TEST 2-

A = [];
count = 1;
[count, A] = updateA('EaglesS1.wav',count,A,30,10);
[count, A] = updateA('EaglesS2.wav',count,A,30,10);
[count, A] = updateA('Journey1.wav',count,A,30,10);
[count, A] = updateA('Journey2.wav',count,A,30,10);
[count, A] = updateA('fleetwood1.wav',count,A,30,10);
[count, A] = updateA('fleetwood2.wav',count,A,30,10);
%A = A'

[UA,SA,VA] = svd(A,0);
VA = VA(:,1:10);
%Create Test Data
B = [];
count = 1;
[count, B] = updateA('EaglesS3.wav',count,A,30,10);
[count, B] = updateA('EaglesS4.wav',count,A,30,10);
[count, B] = updateA('Journey3.wav',count,A,30,10);
[count, B] = updateA('Journey4.wav',count,A,30,10);
[count, B] = updateA('fleetwood3.wav',count,A,30,10);
[count, B] = updateA('fleetwood4.wav',count,A,30,10);
%B = B';

[UB,SB,VB] = svd(B,0);
VB = VB(:,1:10);
% Make the group names
group = [];
for i = 1:20
    group(i) = 1;
end
for i = 21:40
    group(i) = 2;
end
for i = 41:60
    group(i) = 3;
end
group = group';

%%
[class,err,POSTERIOR,logp,coeff] = classify(VB,VA,group,'linear');

VA = VA';
eagles = [];
journey = [];
fleetwood = [];

for i = 1:60
    if (0<i)&& (i<21)
        eagles = [eagles;norm(VA(:,i))];
    elseif (21 <= i)&&(i < 41)
```

```
        journey = [journey;norm(VA(:,i))];
    elseif (41 <= i) && (i<61)
        fleetwood = [fleetwood;norm(VA(:,i))];
    end
end
all = [eagles;journey;fleetwood];

%%
figure(4)
subplot(2,1,1)
plot(group,'bo')
title('True Group Labels of Training Data')
xlabel('Training Group Sample')
ylabel('label')
subplot(2,1,2)
plot(class,'ro')
title('Classification of Test Data')
xlabel('Training Group Sample')
ylabel('label')

%%
%calculate error
corr = 0;
for i = 1:length(group)
    if group(i) == class(i)
        corr =corr+ 1;
    end
end
error = (length(group)-corr)/(length(group))*100;
disp(error);

%%
figure(5)
plot(eagles,'bo')
hold on
plot(journey,'ro')
hold on
plot(fleetwood,'go')
hold on
%% TEST 3-
A = [];
count = 1;
[count, A] = updateA('classical.wav',count,A,30,30);
[count, A] = updateA('classical2.wav',count,A,30,30);
[count, A] = updateA('EaglesS1.wav',count,A,30,10);
[count, A] = updateA('EaglesS2.wav',count,A,30,10);
[count, A] = updateA('Journey1.wav',count,A,30,10);
[count, A] = updateA('Journey2.wav',count,A,30,10);
[count, A] = updateA('fleetwood1.wav',count,A,30,10);
[count, A] = updateA('fleetwood2.wav',count,A,30,10);
[count, A] = updateA('Snoop1.wav',count,A,30,10);
[count, A] = updateA('Snoop2.wav',count,A,30,10);
[count, A] = updateA('Eminem1.wav',count,A,30,10);
[count, A] = updateA('Eminem2.wav',count,A,30,10);
```

```
[count , A] = updateA('50cent1.wav',count,A,30,10);
[count , A] = updateA('50cent2.wav',count,A,30,10);

%A = A'

[UA,SA,VA] = svd(A,0);
VA = VA(:,1:10);
%Create Test Data
B = [];
count = 1;
[count , B] = updateA('classical3.wav',count,A,30,30);
[count , B] = updateA('classical4.wav',count,A,30,30);
[count , B] = updateA('EaglesS3.wav',count,A,30,10);
[count , B] = updateA('EaglesS4.wav',count,A,30,10);
[count , B] = updateA('Journey3.wav',count,A,30,10);
[count , B] = updateA('Journey4.wav',count,A,30,10);
[count , B] = updateA('fleetwood3.wav',count,A,30,10);
[count , B] = updateA('fleetwood4.wav',count,A,30,10);
[count , B] = updateA('Snoop3.wav',count,A,30,10);
[count , B] = updateA('Snoop4.wav',count,A,30,10);
[count , B] = updateA('Eminem3.wav',count,A,30,10);
[count , B] = updateA('Eminem4.wav',count,A,30,10);
[count , B] = updateA('50cent3.wav',count,A,30,10);
[count , B] = updateA('50cent4.wav',count,A,30,10);
%B = B'

[UB,SB,VB] = svd(B,0);
VB = VB(:,1:10);
% Make the group names
group = [];
for i = 1:60
    group(i) = 1;
end
for i = 61:120
    group(i) = 2;
end
for i = 121:180
    group(i) = 3;
end
group = group';

%%
[class,err,POSTERIOR,logp,coeff] = classify(VB,VA,group,'linear');

VA = VA';
classical = [];
rock = [];
rap = [];

for i = 1:60
    if (0<i)&& (i<61)
        classical = [classical;norm(VA(:,i))];
    elseif (61 <= i)&&(i < 121)
        rock = [rock;norm(VA(:,i))];
```

```
    elseif (121 <= i) && (i<181)
        rap = [rap;norm(VA(:,i))];
    end
end
all = [classical;rock;rap];

%%
figure(4)
subplot(2,1,1)
plot(group,'bo')
title('True Group Labels of Training Data')
xlabel('Training Group Sample')
ylabel('label')
subplot(2,1,2)
plot(class,'ro')
title('Classification of Test Data')
xlabel('Training Group Sample')
ylabel('label')

%%
%calculate error
corr = 0;
for i = 1:length(group)
    if group(i) == class(i)
        corr =corr+ 1;
    end
end
error = (length(group)-corr)/(length(group))*100;
disp(error);

%%
figure(5)
plot(classical,'bo')
hold on
plot(rock,'ro')
hold on
plot(rap,'go')
hold on

%% Functions:
function[yspec] = makeSpectrogram(y,Fs)
    N = length(y); % sample lenth
    slength = N/Fs; % total time span of audio signal
    t = linspace(0, N/Fs, N);
    %plot(t, y); % pplots the audio
    L = t(end);
    k = (1/L)*[0:(N/2)-1 -(N/2):-1]; ks = fftshift(k);
    y = y';
    yspec=[];
    tslide=0:0.5:slength;
    for j=1:length(tslide)
        g=exp(-(100)*(t-tslide(j)).^2);
        yg=g.*y;
        ygt=fft(yg);
```

```
        yspec=[yspec;abs(fftshift(ygt))];
    end
end

function [newc, newA] = updateA(file,count,A,start, numcol)
    [y,Fs] = audioread(file);
    for i = start :5:((start-5)+(5*numcol))
        sample = [i*Fs,(i+5)*Fs];
        [muz,fs] = audioread(file, sample);
        muz = muz(:,1);
        muz = muz(1:5:length(muz));
        fs = fs/5;
        sp = makeSpectrogram(muz,fs);
        n = length(sp(1,:));
        m = length(sp(:,1));
        spv = reshape(sp,n*m,1);
        A(:,count) = spv;
        count = count+1;
    end
    newA = A;
    newc = count;
end
```