AMATH 582 Homework 4

Christopher Liu

March 6, 2020

Abstract

SVD analysis and linear discriminant analysis are two powerful tools that allow us to extract key features from data. In this assignment we will be applying the SVD analysis to analyze 2 sets of face portraits. Using the SVD analysis, we will determine the proper orthogonal modes and compute rank approximations of the faces. A comparison of the uncropped and cropped data will then highlight potential drawbacks when the data is not centered. For the second part, we will use linear discriminant analysis to create an algorithm capable of distinguishing music from different bands and genres. The SVD is also used to identify key features from spectrograms of our music samples that can then be used to train our algorithm. We will then observe how our algorithm performs with high error due to the limited training data and choice of bands and genres.

1 Introduction and Overview

In this assignment, we will be exploring 2 different types of analyses, SVD analysis and linear discriminant analysis. For the first part, we will be applying SVD analysis to 2 sets of faces. One set has had the photos cropped and centered while the other has not. By using the SVD analysis, we will explore the concept of eigenfaces and how they can be used to determine the number of dominant modes needed to approximate a face. We will then compare the result of the SVD analysis for the uncropped and cropped data and observe any differences.

For the second part, we will be applying linear discriminant analysis to music samples to create an algorithm that can classify a given piece of music by sampling a 5 second clip. To test our algorithm, we will do 3 tests. For the first test, we test whether the the algorithm can identifying music from 3 different bands of different genres. For the second test, we test whether the algorithm can correctly identify music from 3 different bands of the same genre. Finally, for the last test, we will test the algorithm to see if it can distinguish music from different bands for 3 distinct genres. In order to train the algorithm and test data, we will compute and analyze spectrograms of the music to identify characteristics in the frequency and time of a given band or genre.

2 Theoretical Background

2.1 Singular Value Decomposition

We once again employ the SVD and the related covariance matrix to analyze the faces. Every matrix $A \in^{m\times n}$ has a singular value decomposition (SVD) given by Equation (1) [1].

$$A = U\Sigma V^* \tag{1}$$

where $U \in^{m\times m}$ is unitary, $V \in^{n\times n}$ is unitary and $\Sigma \in^{m\times n}$ is diagonal. Furthermore, σ_j, the components of Σ are uniquely determined for all A. The columns of U contain the linear proper orthogonal modes (POD) which constitute the orthonormal expansion basis of interest, Σ determines the strength of each projection onto the POD modes and the columns of V^T show how X is projected onto the new basis. Additionally, once we calculate the SVD of A, we can compute low rank approximations (2).

$$A_n = \Sigma_{i=1}^{n} \sigma_i U_i V_i^T \tag{2}$$

where A_n is the nth rank approximation of A and $U_i,\ V_i$ are the ith columns of U and V respectively.

2.2 Linear Discriminant Analysis

Once the time-frequency information has been extracted and the SVD has been computed to determine the proper orthogonal modes of the data, we will employ linear discriminant analysis (LDA) to make a statistical decision based on the data's PODs. Proposed by Fisher in the context of taxonomy [1], the idea of LDA is to find a suitable projection that maximizes distance between data of different classes while minimizing the distance between data of the same class. Using training data, the LDA will determine a decision threshold level that will be used to classify new data. The more dominant modes and training data included, the more accurate the decision threshold will be, but at the cost of computational time.

3 Algorithm Implementation and Development

3.1 Yale Faces: SVD Analysis

There are 2 main steps in performing the SVD analysis on the faces. First, we need to load the data from the different sub-directories into a single matrix A which we will then decompose. It is important that any bad or corrupt data is omitted as it can skew our SVD. We then read in our images which are matrices where elements correspond to a pixel and the value of that element is its intensity in black and white. Each matrix corresponding to an image is reshaped into a column vector and A is then a matrix whose columns are each individual image. Finally, we subtract the row mean of our matrix A in order to 'center' our intensities. For the second step, we compute the reduced SVD of A. Note that in order to explore the resulting data, we will have to reshape our columns of U to plot them.

3.2 Music Classification: LDA

Similar to Part 1, there are 2 main steps we take to perform LDA on our music. First, we must load all the data we need and extract the dominant modes from the time-frequency spectra of the songs. Next, we will use a function in MATLAB to perform the LDA on our training and test data.

Loading in the audio, we find that we have 2 columns of sampled data corresponding to the left and right channel. For this assignment, we choose to use the first column. Next, we need to downsample the music to reduce the computational time. A 5 second interval is randomly chosen in the song and only every 100th element in the resulting vector is used. Once we have our downsampled 5 second sample, we perform a Gabor transform to produce a spectrogram (See section 3 in Assignment 2).These are then reshaped into a column vector and stored in a matrix corresponding to a band or genre. Once the spectrograms for a given band or genre have been reshaped and stored in a matrix, it is exported so that it can be loaded for later use without having to recompute it. For each given test, we combine all of the data into one matrix and compute its SVD.

Each row of V corresponds to the POD of each corresponding music sample. We can then use rank approximations (the first n columns) to create training and testing data sets. Once an appropriate approximation has been determined, we input our approximated data into the MATLAB function `classify` which will perform the LDA and test it on specified data.

4 Computational Results

4.1 Yale Faces

4.1.1 Cropped Faces

Computing the the SVD of the yale faces, we observe that there are a few dominate modes (Figure 1). We can plot different rank approximations of the first cropped face and observe how many ranks we need to get a decent approximation (Figure 2). Note that this is the de-meaned data so it is darker than normal. Comparing to the original, we note that rank 50 gives us a decent approximation albeit a little blurry. The first 50 modes captures approximately 34.6% of the energy. We should also note that the large negative singular values skew the cumulative energy..

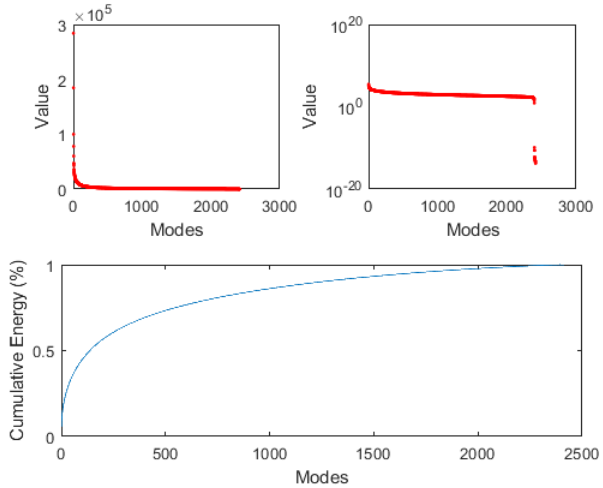

Figure 1: The singular values and cumulative energy for cropped faces

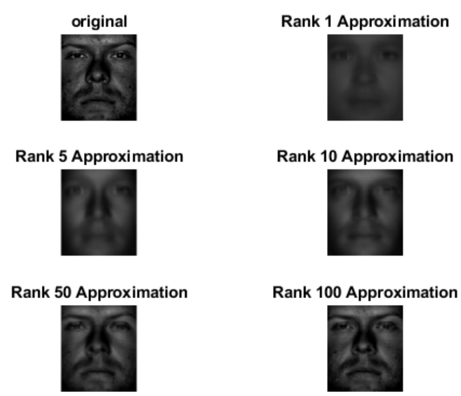

Figure 2: Rank approximations of the 1st cropped face

Plotting the first 4 columns of U, we find that this is the corresponding 'eigenface' basis of our cropped faces. So U is our new basis and Σ is the strength of the projection (or weights) of each face onto the POD modes. Any given face, a column of A, can be represented as a linear combination of the weighted eigenfaces and the coeffiecients of the linear combination are determined by a column of V^T. That is, the first cropped face, A_1, is given by, $A_1 = U\Sigma V_1^T$, where V_1 is the first column of V^T

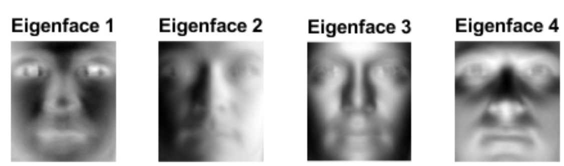

Figure 3: Eigenface basis for cropped faces

4.1.2 Uncropped Faces

Moving onto the uncropped faces, we notice that the SVD A produces similar results to the cropped case however we do not have have very large singular values that are an order larger anymore (Figure 4). Looking again at the rank approximations for the 1st uncropped faces, we note that we get a pretty good approximation at rank 50 which is about 69.5% of the energy (Figure 5). This is much higher since the negative modes have smaller values.

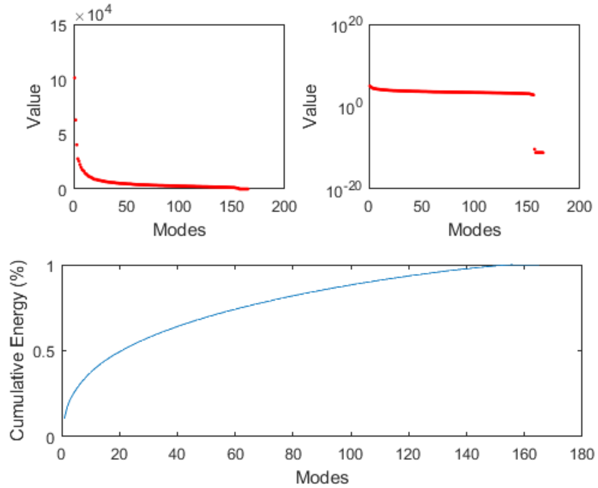

Figure 4: he singular values and cumulative energy for uncropped faces

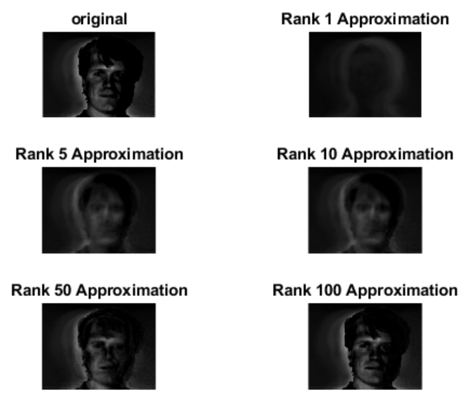

Figure 5: Rank approximations of the 1st uncropped face

Turning our attention to the columns of U (Figure 6), we notice a significant difference between the uncropped and cropped images. Because the face is not centered in the uncropped data set, we observe that our columns now show multiple faces unlike the cropped data set. This is because our basis now has to be able to re-create portraits where the face can be anywhere in the image.

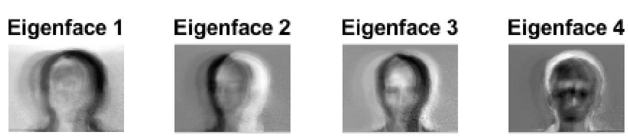

Figure 6: Eigenface basis for uncropped faces

4.2 Music Classification

For all of the music classification, we used a Gaussian Gabor window with an increment of 0.2 over the 5 second sample and a width of 50. 10 samples for each band/genre are used to train and another 10 for each band/genre are used to test. A rank 10 approximation is used for all 3 cases.

4.2.1 Test 1: Band Classification

For Test 1, we use 3 bands of different genres, AC/DC, Daft Punk and Jake Bugg. Our classification does an okay job with a success rate of 60%. The rank 10 approximation captures 37% of the energy.

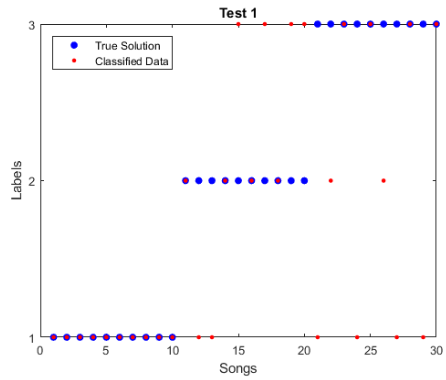

Figure 7: LDA Classification Results for Test 1

4.2.2 Test 2: The Case for Seattle

For Test 2, we use 3 different pop punk bands, Green Day, Yellowcard and No Use for a Name. Surprisingly, our LDA classification has a success rate of 63%. The rank 10 approximation captures 36% of the energy.

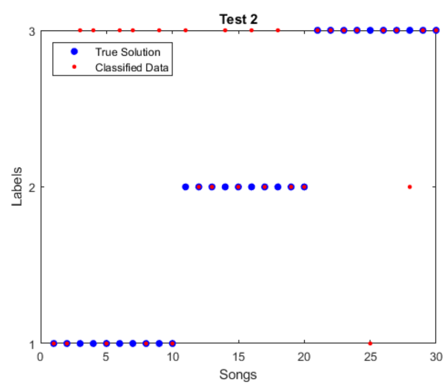

Figure 8: LDA Classification Results for Test 2

4.2.3 Test 3: Genre Classification

For Test 3, we use a variety of bands from 3 genres, pop punk, korean pop and rap. Our classifcation algorithm has the lowest success rate with only 50% in this test. The rank 10 approximation captures 36% of the energy.

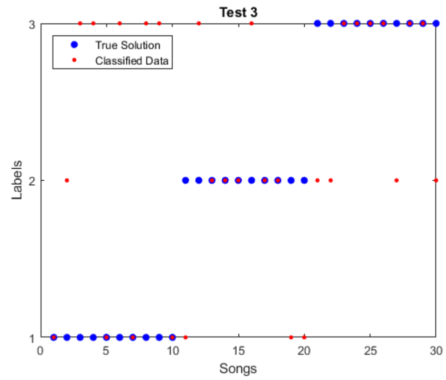

Figure 9: LDA Classification Results for Test 3

5 Summary and Conclusions

For the 1st part, we are able to demonstrate the power of SVD analysis to extract the identifying characteristics of a data set (the eigenfaces). We were able to determine a new basis to project the data on and also which directions in this new basis are the dominant directions. As a result, we can use the SVD analysis to approximate images and represent the same information almost exactly but without having to store all the data. A comparison of the uncropped and cropped dataset highlights shortcomings of the SVD analysis when your data in the set isn't all centered.

The music classification produced some interesting unexpected results. We would have expected Test 1 and 3 to perform significantly better than Test 2 but Test 2 ended up being marginally better. It is important to note that we have limited training data (only 30 samples for each test from 3 bands/genres) and that we are only comparing the time-frequency component of each sample. Certain bands may have similar spectrograms despite being different genres. In the case of Test 2, the 3 bands I chose may have been distinct enough despite being labelled as the same genre. For example, Yellowcard has a violinist while the other 2 bands do not. The samples were also only 5 second clips of each song and were significantly down sampled. A similar argument can be made for Test 3 since rap and k-pop are broad genres with many different styles that could overlap with the other genres.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data.* Oxford University Press, 2013.

Appendix A MATLAB Functions

Below are the key MATLAB functions implemented.

- `dir` lists files and folders int he current folder
- `Y=reshape(A,X,1)` reshapes matrix A into a X by 1 vector
- `imshow(U)` displays the picture given by the intensity values in a figure
- `imread(filename)` reads in an imagine specified by the filename
- `[U,S,V]=svd(A,0)` returns the reduced singular value decomposition of A
- `[y, Fs]=audioread(filename)` reads data from the specific file and returns the sampled data, `y`, and the sample rate, `Fs`
- `fft(Vg)` returns the multidimentional Fourier transform of an n-dimensional array by using the fast Fourier Transform algorithm.

- `fftshift(Vgt)` rearranges the Fourier transform so that the zero-frequency component is at the center. Used for plotting transformed data.
- `classify(sample,training,group)` performs LDA classifcation on the training data that is labeled by group. It then tests the resulting decision threshold on the sample data.
- `randperm(n)` returns a random permuation of integers from 1 to n without repeating elements.

Appendix B MATLAB Code

https://github.com/chrismhl/AMATH582/tree/master/Homework4

B.1 Yale Faces

```
clear all; clc; close all;

% Importing the cropped face
D = 'C:\XXX';
S = dir(fullfile(D,'*'));
N = setdiff({S([S.isdir]).name},{'.','..'}); % list of subfolders of D.
index = 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(F,'pgm');
        X_f(:,index) = reshape(A,[32256,1]);
        index = index+1;
    end
end
%%
% Subtract average from the face for cropped
X_f_avg = zeros(192*168, 2414);

for i = 1:192*168
    X_f_avg(i,:) = X_f(i,:) - mean(X_f(i,:));
end
%%
% SVD for cropped images
[Uc,Sc,Vc] = svd(double(X_f),0);
[Uc_a,Sc_a,Vc_a] = svd(X_f_avg,0);

%%
% Plot singular values

dSc = diag(Sc);
dSc_a = diag(Sc_a);

figure(1)
subplot(2,2,1)
plot(1:length(Sc),dSc,'.r');
xlabel('Modes')
ylabel('Value')
subplot(2,2,2)
semilogy(1:length(Sc),dSc,'.r');
xlabel('Modes')
```

```
ylabel('Value')

%calculate energy
energy = zeros(length(dSc),1);
e_total = sum(dSc);
for i = 1:length(dSc)
   energy(i)= sum(dSc(1:i))/e_total;
end

subplot(2,2,[3,4])
plot(1:length(dSc),energy)
xlabel('Modes')
ylabel('Cumulative Energy (%)')

%mean subtracted
figure(2)
subplot(2,2,1)
plot(1:length(Sc_a),dSc_a,'.r');
xlabel('Modes')
ylabel('Value')
subplot(2,2,2)
semilogy(1:length(Sc_a),dSc_a,'.r');
xlabel('Modes')
ylabel('Value')

%calculate energy
energy_a = zeros(length(dSc_a),1);
e_a_total = sum(dSc_a);
for i = 1:length(dSc_a)
   energy_a(i)= sum(dSc_a(1:i))/e_a_total;
end

subplot(2,2,[3,4])
plot(1:length(dSc_a),energy_a)
xlabel('Modes')
ylabel('Cumulative Energy (%)')
%%
% Rank approximation to the faces.
rank = [1 5 10 50 100];
figure(3)
subplot(3,2,1)
imshow(reshape(uint8(X_f_avg(:,1)),[192,168]))
title('original')

for k = 1:length(rank)
    Xc_approx = Uc_a(:,1:rank(k))*Sc_a(1:rank(k),1:rank(k))*Vc_a(:,1:rank(k))';
    C = reshape(uint8(Xc_approx(:,1)),[192,168]);
    subplot(3,2,k+1)
    imshow(C);
    title(['Rank ', num2str(rank(k)), ' Approximation'])
end

%%
% Eigenfaces for cropped
j = 4;
for k = 1:j
    maxv = max(Uc_a(:,k));
```

```
    minv = min(Uc_a(:,k));

    figure(10)
    subplot(1,j,k)
    imshow(reshape(Uc_a(:,k),[192,168]),[minv maxv])
    title(['Eigenface ', num2str(k)])
end

%%
% Importing the uncropped faces
D = 'C:\XXX';
S = dir(fullfile(D,'*'));

C = {S(~[S.isdir]).name}; % files in subfolder.
for jj = 1:numel(C)
    F = fullfile(D,C{jj});
    A = imread(F);
    dim = size(A);
    X_f_full(:,jj) = reshape(A,[dim(1)*dim(2),1]);
end

%%
% Subtract average from the face for uncropped
X_f_full_avg = zeros(77760, 165);

for i = 1:77760
    X_f_full_avg(i,:) = X_f_full(i,:) - mean(X_f_full(i,:));
end
%%
% SVD for uncropped images
[Uf,Sf,Vf] = svd(double(X_f_full),0);
[Uf_a,Sf_a,Vf_a] = svd(X_f_full_avg,0);

%%
% Plot singular values

dSf = diag(Sf);
dSf_a = diag(Sf_a);

figure(4)
subplot(2,2,1)
plot(1:length(Sf),dSf,'.r');
xlabel('Modes')
ylabel('Value')
subplot(2,2,2)
semilogy(1:length(Sf),dSf,'.r');
xlabel('Modes')
ylabel('Value')

%calculate energy
energy = zeros(length(dSf),1);
e_total = sum(dSf);
for i = 1:length(dSf)
   energy(i)= sum(dSf(1:i))/e_total;
end

subplot(2,2,[3,4])
```

```
plot(1:length(dSf),energy)
xlabel('Modes')
ylabel('Cumulative Energy (%)')

%mean subtracted
figure(5)
subplot(2,2,1)
plot(1:length(Sf_a),dSf_a,'.r');
xlabel('Modes')
ylabel('Value')
subplot(2,2,2)
semilogy(1:length(Sf_a),dSf_a,'.r');
xlabel('Modes')
ylabel('Value')

%calculate energy
energy_a = zeros(length(dSf_a),1);
e_a_total = sum(dSf_a);
for i = 1:length(dSf_a)
   energy_a(i)= sum(dSf_a(1:i))/e_a_total;
end

subplot(2,2,[3,4])
plot(1:length(dSf_a),energy_a)
xlabel('Modes')
ylabel('Cumulative Energy (%)')
%%
% Rank approximation to the faces (full).
rank = [1 5 10 50 100];
figure(6)
subplot(3,2,1)
imshow(reshape(uint8(X_f_full_avg(:,1)),[243,320]))
title('original')

for k = 1:length(rank)
    Xf_approx = Uf_a(:,1:rank(k))*Sf_a(1:rank(k),1:rank(k))*Vf_a(:,1:rank(k))';
    C = reshape(uint8(Xf_approx(:,1)),[243,320]);
    subplot(3,2,k+1)
    imshow(C);
    title(['Rank ', num2str(rank(k)), ' Approximation'])
end

%%
j = 4;
for k = 1:j
    maxv = max(Uf_a(:,k));
    minv = min(Uf_a(:,k));

    figure(10)
    subplot(1,j,k)
    imshow(reshape(Uf_a(:,k),[243,320]),[minv maxv])
    title(['Eigenface ', num2str(k)])
end
```

B.2 Music Classification

```
clear all; clc; close all;

```

```
%%
% Importing music, creating spectrograms and ouputting as .mat
% This was done to make it easier to move data between laptop and desktop
D = 'C:\XXX'; %change as needed
S = dir(fullfile(D,'*.wav'));
X = [];

C = {S(~[S.isdir]).name}; % files in folder.
for jj = 1:10
    start = randi([15,25]); %change as needed
    F = fullfile(D,C{jj});
    [y, Fs] = audioread(F);

    spec=[];
    incr = 0.2;
    tslide_p=0:incr:5;
    width = 50; %change as needed
    yc = y(start*Fs:100:(start+5)*Fs,1);
    if mod(length(yc),10) == 1
        yc = yc(1:length(yc)-1);
    end
    n = length(yc);

    t2=linspace(0,5,n+1);
    t2 = t2(1:n);
    kp=(1/5)*[0:n/2-1 -n/2:-1];
    ksp=fftshift(kp);


    for j=1:length(tslide_p)
        g=exp(-width*(t2-tslide_p(j)).^2); % Gabor
        Pg=g.*yc.';
        Pgt=fft(Pg);
        spec=[spec; abs(fftshift(Pgt))];
    end

    dim = size(spec);
    x = reshape(spec,[dim(1)*dim(2), 1]);

    X = [X, x];
end
dp = X;
save('dp.mat','dp');

%%
% Load data and perform SVD for Test 1
clear all; clc; close all

load acdc.mat
load acdc2.mat
load dp.mat
load dp2.mat
load jb.mat
load jb2.mat

A = [acdc, acdc2, dp, dp2, jb, jb2];

```

```
[U, S, V] = svd(A,0);

test = [V(1:5,1:10); V(11:15,1:10); V(21:25,1:10); V(31:35,1:10);V(41:45,1:10); V(51:55,1:10)];
train = [V(6:10,1:10); V(16:20,1:10); V(26:30,1:10); V(36:40,1:10);V(46:50,1:10); V(56:60,1:10)];

group = ones(30,1);
group(1:10) = 1.*group(1:10);
group(11:20) = 2.*group(11:20);
group(21:30) = 3.*group(21:30);

[class, err, logp, coeff] = classify(test, train, group);

correct =0;
for i = 1:length(class)
    if class(i) == group(i)
        correct = correct +1;
    end
end
error_1 = correct/length(group);

figure(1)
plot(group,'b.','MarkerSize',20);
hold on
plot(class,'r.','MarkerSize',10);
hold off
yticks([1,2,3])
legend('True Solution','Classified Data','Location','NorthWest')
title('Test 1')
xlabel('Songs')
ylabel('Labels')
%%
% Load data and perform SVD for Test 2
close all; clear all; clc;

load gd1.mat
load gd2.mat
load yc1.mat
load yc2.mat
load nufan1.mat
load nufan2.mat

A = [gd1,gd2,yc1,yc2,nufan1,nufan2];

[U, S, V] = svd(A,0);

test = [V(1:5,1:10); V(11:15,1:10); V(21:25,1:10); V(31:35,1:10);V(41:45,1:10); V(51:55,1:10)];
train = [V(6:10,1:10); V(16:20,1:10); V(26:30,1:10); V(36:40,1:10);V(46:50,1:10); V(56:60,1:10)];

group = ones(30,1);
group(1:10) = 1.*group(1:10);
group(11:20) = 2.*group(11:20);
group(21:30) = 3.*group(21:30);

```

```
[class, err, logp, coeff] = classify(test, train, group);

correct =0;
for i = 1:length(class)
    if class(i) == group(i)
        correct = correct +1;
    end
end
error_2 = correct/length(group);

figure(2)
plot(group,'b.','MarkerSize',20);
hold on
plot(class,'r.','MarkerSize',10);
hold off
yticks([1,2,3])
legend('True Solution','Classified Data','Location','NorthWest')
title('Test 2')
xlabel('Songs')
ylabel('Labels')
%%
% Load data and create data and perform SVD for Test 3
close all; clear all; clc;

load gd1.mat
load gd2.mat
load yc1.mat
load yc2.mat
load nufan1.mat
load nufan2.mat
load bob.mat
load kpop.mat
load b182.mat
load djdeck.mat
load ok.mat

punk = [gd1,yc1,nufan1,b182];
rap = [bob, djdeck, ok];

pperm = randperm(size(punk,2),20);
rperm = randperm(size(rap,2),20);
kperm = randperm(size(kpop,2),20);

A = [punk(:,pperm),rap(:,rperm),kpop(:,kperm)];
[U, S, V3] = svd(A,0);

save('V3.mat','V3');
save('S3.mat','S');
%%
% Classify test 3
close all; clear all; clc;
load V3.mat
V = V3;

train = [V(1:5,1:10); V(11:15,1:10); V(21:25,1:10); V(31:35,1:10);V(41:45,1:10); V
    (51:55,1:10)];
test = [V(6:10,1:10); V(16:20,1:10); V(26:30,1:10); V(36:40,1:10);V(46:50,1:10); V
```

```
    (56:60,1:10)];

group = ones(30,1);
group(1:10) = 1.*group(1:10);
group(11:20) = 2.*group(11:20);
group(21:30) = 3.*group(21:30);

[class, err, logp, coeff] = classify(test, train, group);

correct =0;
for i = 1:length(class)
    if class(i) == group(i)
        correct = correct +1;
    end
end
error_3 = correct/length(group);

figure(3)
plot(group,'b.','MarkerSize',20);
hold on
plot(class,'r.','MarkerSize',10);
hold off
yticks([1,2,3])
legend('True Solution','Classified Data','Location','NorthWest')
title('Test 3')
xlabel('Songs')
ylabel('Labels')
%%
%Plot singular values and energy
dS = diag(S);
figure(1)
plot(1:length(dS), dS)

%calculate energy
energy = zeros(length(dS),1);
e_total = sum(dS);
for i = 1:length(dS)
   energy(i)= sum(dS(1:i))/e_total;
end
figure(2)
plot(1:length(dS),energy)


%%
% Test code for the spectrogram. delete later if needed
pcolor(tslide_p,ksp,(spec.'/max(max(abs(spec)))))
shading interp
ylim([0 max(ksp)])
set(gca,'Fontsize',[14])
colormap(hot)
```