AMATH 482 HW 4

Liyuan Tang

March 2019

Abstract

In the first part of this assignment, I will load the data of the original images and cropped images. Then, I will do the SVD analysis of those data sets and compare the results. Also, I will find the rank of the face space in order to have a good reconstruction. For the music classification, I will use different songs to construct the data input. Then, I will use that matrix to do the SVD and create a training set and test set. For each test case, I will use different algorithm to identify the test set and calculate the accuracy.

1 Introduction

We are given two data sets which are the cropped images and cropped images. For each data set, I will use a for loop to load the data and store the images in the matrix columns by columns. Then, I will do the SVD analysis and compare the difference in the singular value spectrum between these two data sets. I will determine the rank for the face space in order to truncate results from SVD and reconstruct the original matrix. For the music classification, the first test is three different bands from different genres. I chose AC/DC, Drake and Johann Sebastian Bach. The genres covered are rock, hip hop and classic. The second case I choose Alice In Chains, Pearl Jam and Soundgarden which are three rock bands. The last case is various bands with different genres. I limited my result to three genres, rock, classic and rock. I used 6 songs for the first two test, two for each. And I used 18 songs in the last test, which each genre I chose three artists and for each artists I used two songs. I took 5 second clips as the sample, took their spectrogram and stored them in a matrix where each column is a different sample. Then, I did the SVD and randomly assigned the samples to the training set and test set. I used Naive Bayes, k-Nearest Neighbors and LDA to classify the data several times and calculate the average accuracy for each algorithm on each test.

2 Theoretical Background

2.1 Singular Value Decomposition

The singular value decomposition is a factorization of matrix into a number of constitutive components all of which have a specific meaning in applications [1]. Based on the theorem in the notes, SVD could diagonalize all matrices and provide their singular value and orthogonal bases. It is a transformation that stretches or compresses a given set of vector and then rotate it. The representation of full SVD is:

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

The size of A is m × n. And U is an m × m unitary matrix, with orthogonal columns. Σ is an m × n diagonal matrix. And V is an n × n unitary matrix. Besides, the diagonal elements of Σ are nonnegative and sorted from largest to smallest. Thus, the SVD of the matrix A shows that the matrix first uses a unitary transformation which is a rotation through V^*. Then, it applies a stretching operation by Σ. Finally, it rotates again by the unitary transformation U.

2.2 Clustering and classification

The goal of machine learning is to find a low-rank subspace for optimizing the data and regression methods for clustering and classification of different data types. There are two categories for machine learning which are the supervised and unsupervised machine learning. In supervised machine learning, algorithm is used for labelled data sets. And in this homework, I will use three supervised learning methods which are k-nearest neighbors, Naive Bayes and linear discriminant analysis.

k-nearest neighbors(kNN): This is a supervised algorithm. It means that when given a new data point x_k which does not have a label, it will simply find the k nearest neighbors x_j with labels y_j. The label of the new point x_k is determined by a majority vote of the kNN. In Matlab, when we have a model of data, we can use kNN by: label = knnsearch(Mdl,test, 'k', n) which find n nearest neighbors of test, which is the new data input in the Mdl, which is the data with labels.

Naive Bayes: The Naive Bayes algorithm provides an intuitive framework for supervised learning[1]. It's easy to construct and is similar to SVM which doesn't need complicated parameter estimation. It is built on Baye's theorem and the conditional probability with the equation as :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

It allows people to estimate the label of test data based on the prior probability distributions of the labeled training data.

Linear discriminant analysis: LDA enables different labeled data have clear separation between their distribution of points. The goal of LDA is to find a suitable projection that maximize the distance between the inter-class data while minimizing the distance between intra-class data[1]. With the optimization, LDA separates the probability distribution functions for each class in an optimal way.

3 Algorithm Implementation and Development

3.1 Yale Faces B

First, I extracted all the file from the zipped folder. Then, I used a for loop to load the cropped data. I iterated from 1 to 39, since "yaleB14" doesn't exist in this data, I add another if operation under the for loop that would jump over 14 which is "if i ∼= 14". Since my Matlab file is in the same folder as "yalefaces_cropped", I created the path to get the final folder which is "file = './yalefaces_cropped/CroppedYale/yaleB' ". Then I concatenate this path with the number of iteration to get the full path to each folder that contains the files. One thing to notice is that when the number is less than 10, we should add a 0 before the number such as "yaleB08" instead of "yaleB8". Then, I used "list = dir(final)" to list the folder contents and used another for loop to extract all files in this folder. When we call "list.name", the first two item will always be "*" and "**" which are the wildcards. As a result, when using a for loop, we should start from the third index, that is "for j = 3:66". I generated the path to the file, load the image file and change it to double precision. Then, I reshape it to a column vector whose size is 32256 × 1 and store in the matrix A columns by columns. Another thing to notice is since 14 doesn't exist, when we store the data we should notice to change the index after 14. I created a number called "curr", when the iteration number is smaller than 14 curr=0, when greater than 14 curr=1. So when I store the data, I used "A(:,(j-2)+(i-1-curr)*64) = data". Here since j starts from 3, so I need $j-2$, and each folder has 64 files, so I add another (i-1-curr) where i starts from 1 and "curr" which becomes 1 after $i > 14$.

Then, for the uncropped data, I used the same trick which is to create the path to folder that contains 165 files, "direc = './yalefaces_uncropped/yalefaces'". Then, I used "dir" to list all the file names just as I did for the first data set. I used a for loop starts from 1 to 165. During each iteration, I load the data by using "temp = double(imread(strcat(direc,'/', list(i+2).name)))". Here I used $i+2$ since i starts from 1. Then, I reshaped it to a column vector whose size is 77760×1 and stored in the matrix B.

I subtract the mean of A and B for each matrix by mean(A(:)) and then I do the SVD for both matrix. I plot the energy percentage for each node to compare the difference just like I did in the last homework. To find the number of modes that are necessary for good image reconstructions, I tried to find out when the cumulative energy percentage exceeds 90%. I used a for loop to add the energy percentage, once it is greater than 90%, break the loop. I used this index as the rank which I can construct a new u, Σ and v with few size. By multiplying those three matrix together, I can reconstruct the original matrix. And I plotted the reconstructed face 1 for both cropped and original data.

3.2 Music Classification

For test 1, I used two songs from AC/DC which are rock, two songs from Drake which are hip hop and two Bach's songs performed by Tafelmusik Baroque Orchestra which are classical. Just like what I did in part 1, I used a for loop to load the data. For each iteration, I load one song by using "[song, Fs1] = audioread(strcat(path, '/', filename))" which gives me the data "song" and the number of data points per second. I changed the name of each song as "classic 1", "rap 1" and "rock 1" so I can make sure songs in the same genre should be near each other. I took the transpose the data and the data has two rows due to the left and right channels. So I add them up and divided by two to take the average. Then I used a for loop to get several 5 seconds clips on this song by using "for kk = 40:5:160" which takes 25 pieces starts from 40 seconds to 160 seconds. For each clip, I cut the data from song by using "test = song(1, Fs1*kk : Fs1*(kk+5))" which gives me a five second clip. Then, I used "vector = abs(spectrogram(test))" to calculate the absolute value of the short-time Fourier transform of the input signal. Then I reshape it to a row vector to store in the matrix A in rows. After the matrix A was constructed, I took the transpose of A to make data store in columns. Since I have 6 songs and each song I choose 25 piece of 5-second clip, my matrix A will have 150 columns of data.

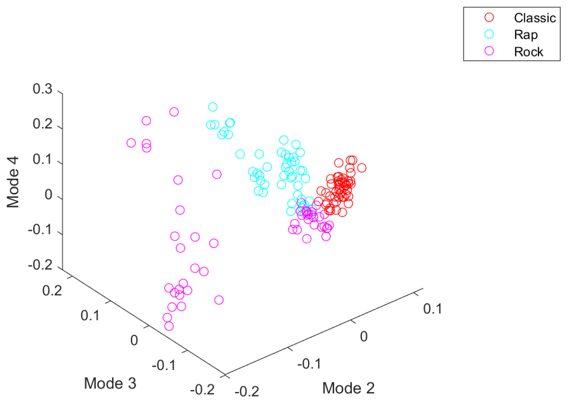
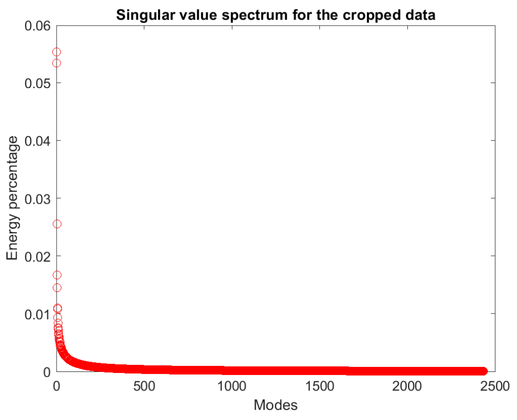
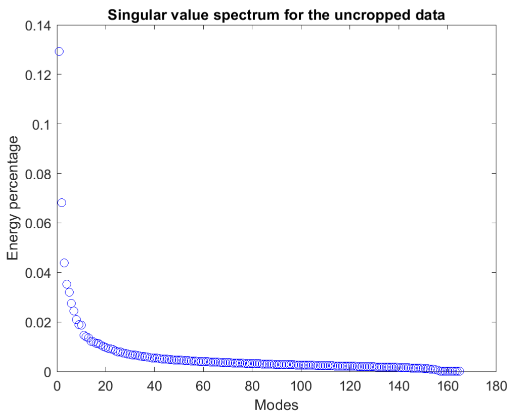


Figure 1: Clusters of the data on 3 modes



(a) Singular value spectrum for the cropped data



(b) Singular value spectrum for the uncropped data

Figure 2: Singular value spectrum for the cropped and uncropped data

Then, I subtracted the mean of A and did the SVD. I used the resulting v to construct my training set and test set. Due to the alphabetical order and the way I load the data, the first 50 rows in v represent the data for classical music. The middle one represent the rap music and the last 50 represent the rock while the columns in v represent the modes. I want to use only three modes so I extracted the data in v by "xclas = v(1:50, 2:4); xpop = v(51:100, 2:4); xrock = v(101:150, 2:4)". I plotted those data in 3d to visualize the data and helped me to determine if I chose good modes (Figure 1). I also create a vector which is the random permutation from 1 to 50, so each time I will have different training set. For training set, I choose 30 rows from the data of each genre. And the row numbers are the first 30 value from the random permutation. Then I used the rest of the data to construct the test set. Then, I apply three algorithms to my data. The first one is k-nearest neighbors (kNN). I just used "ind = knnsearch(xtrain, xtest)" which will return the index in the training set which indicates the nearest neighbor of the data in test set. Then, I changed the output from index to the label which is 1, 2 and 3, 1 represents classical, 2 represents rap and 3 represents rock. For the second algorithm, I did the Naive Bayes. I first created a label consisting of 1,2 and 3 that has the same length as the training set. Then I used "nb = fitcnb(xtrain, label)" to get the model and use it to predict the test set "pre = nb.predict(xtest)". For the last algorithm, I chose LDA which just used Matlab builtin function "classify(test, xtrain, label)". And I repeated each algorithm 100 times for cross validation by changing the random permutation and calculated the mean accuracy.

For test 2 and 3, I just used different songs as the input and did the same thing except changing some parameters.

4 Computational Results

4.1 Yale Face B

1. SVD analysis: See code in Appendix B

2. Interpretation: **U** is an orthonormal basis calculated from the singular value decomposition of A. Each column of U is a principal component. The columns of U are the dominant features of all faces which can be interpreted as the "eigenface". Σ is the singular value, which can be interpreted as the energy for each mode. **V** is also an orthogonal basis. V can be interpreted as the projection of matrix A onto the principal components.

3. Singular value spectrum: Figure 2.

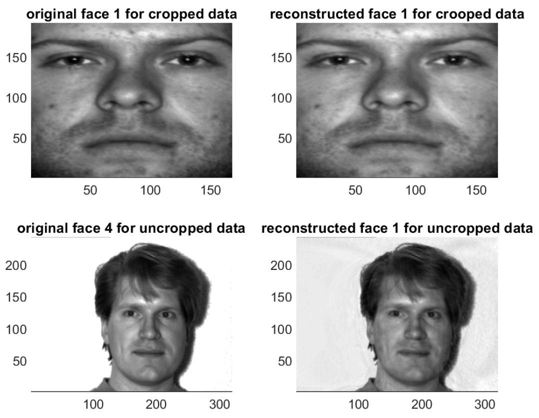

original face 1 for cropped data
reconstructed face 1 for crooped data
original face 4 for uncropped data
reconstructed face 1 for uncropped data

Figure 3: Original face 1 and reconstructed face 1 for cropped and uncropped data. The first row represents the cropped data, the second row represents the uncropped data.

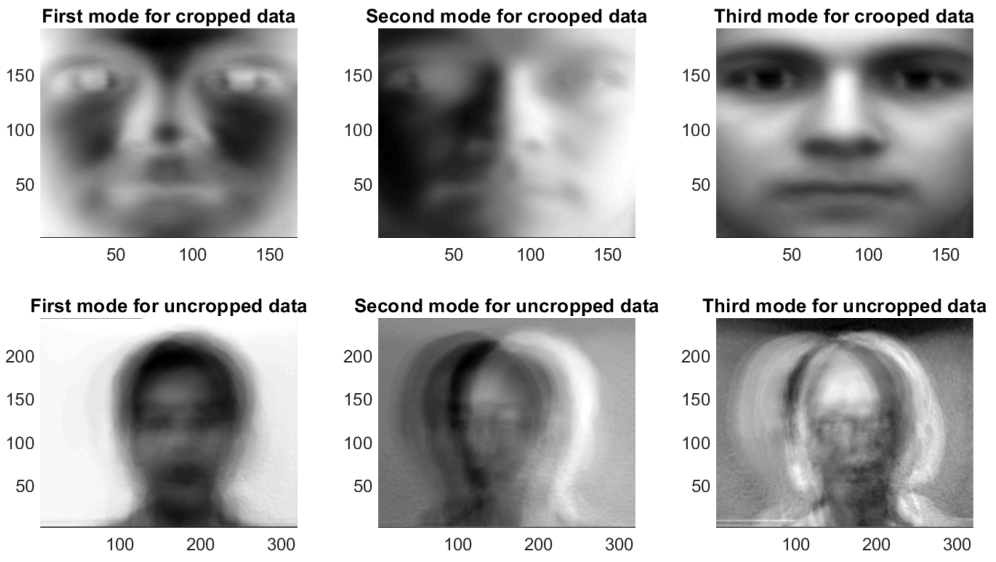

First mode for cropped data
Second mode for crooped data
Third mode for crooped data
First mode for uncropped data
Second mode for uncropped data
Third mode for uncropped data

Figure 4: First three modes for cropped and uncropped data. The first row represents the cropped data, the second row represents the uncropped data.

Based on my implementation, I chose the index where the cumulative energy percentage is greater than 90% to be the rank. So for the cropped data, the rank r of the face space is 1171, and the rank for the uncropped data is 102. I reconstructed the matrix and plotted the first face in the original matrix and reconstructed matrix for both cropped and uncropped data(Figure 3). Also, I plotted the first three modes for both cases(Figure 4).

4. From Figure 2, we can see that the first mode of the uncropped data has higher energy than the first mode of cropped data. This might because in the cropped data, the faces took the whole image while in the original data there are large space of background. Also, the rate of change for cropped data is much higher than the rate of change for original data.

4.2 Music Classification

I used three algorithms to classify the testing set based on the training set. With the resulting labels, we are able to calculate and compare the accuracy of each algorithm on each test. In Figure 5, I plotted

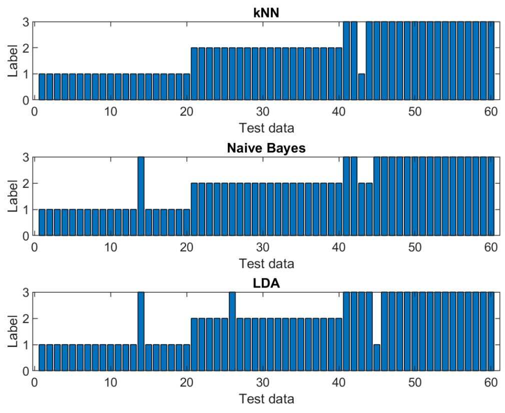


Figure 5: The result of three algorithms on test 1.

Test number	kNN	Naive Bayes	LDA	Average
1	0.9957	0.9645	0.9570	0.9724
2	0.9623	0.9118	0.9450	0.9397
3	0.9769	0.9275	0.8744	0.9263

Table 1: The average accuracy for each algorithm in three tests after runing 100 times

the resulting of three algorithms on test 1 with 1 time running. But due to the cross validation, I ran the whole algorithms 100 times for each test and I printed the average accuracy for each algorithm and the average accuracy for each test after running 100 times in Table 1. We can see on average kNN provides the best accuracy and it almost 100% accurate in the first test. This might because my selected song are highly different which generates different clusters(Figure 1). Since kNN is to find the nearest neighbors, if three genres forms three different clusters which don't overlap, the accuracy in kNN is supposed to be very high. Besides, on average test 1 has the highest accuracy and test 3 has the lowest accuracy. This is reasonable because in the first test, the songs in each genre are from 1 band, but in the third test, the songs in each genre are from different bands.

5 Conclusion

In the first part of this homework, I loaded the data of cropped and uncropped images. Then, I did the SVD on A. With the results from SVD, I plotted the energy percentage for each mode and calculate the rank of the face space which is the number of modes need to have a 90% cumulative energy. By this new rank, I truncated matrix U, Σ and V and reconstructed the image.

The second part is the music classification. There are three different tests. The first one is to compare three artists with different genres, the second compares three artists in a same genre and the last one is different artists in various genres (I limited the results to three genres). I took the spectrogram of each 5-second clip, reshaped it and stored in the matrix A which can be singular value decomposed. Each column of A represents the data of each 5-second clip. After SVD, I used the resulting matrix V to generate the training set and test set. I used kNN, Naive Bayes and LDA to predict the data in the test set based on the training set. For each algorithm in each test, I ran 100 times to cross validate the results and I calculated the average accuracy (Table 1). I think the accuracy is highly depends on the chosen data and it is important to cross validate the results to have a more precise result.

References

[1] J. Nathan Kutz. *AMATH 582 Computation Methods for Data Analysis.* 2010.

6 Appendix A

1. strcat(). It concatenates strings horizontally and returns a string array.

2. dir(name). It returns attributes about name.

3. A = imread(filename). It reads the image from the file specified by filename, inferring the format of the file from its contents.

4. [y, Fs] = audioread(audiofile). It reads data from the file named audiofile, and returns sampled data, y, and a sample rate for that data, Fs.

5. A = spectrogram(x). It returns the short-time Fourier transform of the input signal, x.

6. randperm(n). It returns a row vector containing a random permutation of the integers from 1 to n inclusive.

7. ind = knnsearch(xtrain, test). It finds the nearest neighbor in xtrain for each point in test and returns the indices of the nearest neighbors. ind should be a column vector and have the same number of rows as test.

8. nb = fitcnb(xtrain, label); pre = nb.predict(xtest). The command "fitcnb" will return a multiclass naive Bayes model (Mdl), trained by the predictors in table xtrain and class labels label. Then, you can use this model to predict the labels for the data in xtest.

9. classify(xtest, xtrain, label). It classifies each row of the data in xtest into one of the groups in xtrain which is labeled.

10. bar(data) It creates a bar chart with one bar for each element in data.

7 Appendix B

```
clear all; close all; clc;
%% 1. load cropped data
A = [];
for i = 1:39
    if i ≠ 14  % 14 doesn't exist
      file = './yalefaces_cropped/CroppedYale/yaleB';
       if i < 10
           final = strcat(file,'0',num2str(i));
       else
           final = strcat(file, num2str(i));
       end
       list = dir(final);
       curr = 0;
       if i > 14
           curr = 1;
       end
       for j = 3:66
           temp = double(imread(strcat(final,'/',(list(j).name))));
           A(:,(j-2)+(i-1-curr)*64)=reshape(temp,[192*168,1]);
       end
    end
end
%% 2. load uncropped data
B = [];
direc = './yalefaces_uncropped/yalefaces';
list = dir(direc);
for i = 1:165
    temp = double(imread(strcat(direc,'/', list(i+2).name)));
    B(:, i) = reshape(temp, [243*320, 1]);
end
%% SVD
[u1, s1, v1] = svd(A - mean(A(:)), 'econ');
[u2, s2, v2] = svd(B - mean(B(:)), 'econ');
%% cropped
figure(1)
plot(diag(s1) ./ sum(diag(s1)) , 'ro'); %hold on;
curr = diag(s1) ./ sum(diag(s1));
xlabel('Modes')
ylabel('Energy percentage')
title('Singular value spectrum for the cropped data')
% uncropped
figure(2)
plot(diag(s2) ./ sum(diag(s2)) , 'bo');
curr = diag(s2) ./ sum(diag(s2));
xlabel('Modes')
ylabel('Energy percentage')
title('Singular value spectrum for the uncropped data')
%% Reconstruct A
testdata = diag(s1) ./ sum(diag(s1));
a = 0;
for i = 1:length(s1)
    a = a + testdata(i);
    if (a > 0.9)
        break
    end
```

```
end
rank1 = i;
% Reconstruct B
testdata = diag(s2) ./ sum(diag(s2));
a = 0;
for i = 1:length(s2)
    a = a + testdata(i);
    if (a > 0.9)
        break
    end
end
rank2 = i;
%% plot for original and turncated
newU2 = u2(:, 1:rank2);
newS2 = s2(1:rank2, 1:rank2);
newV2 = v2(:, 1:rank2);
newB = newU2*newS2*newV2';
newU1 = u1(:, 1:rank1);
newS1 = s1(1:rank1, 1:rank1);
newV1 = v1(:, 1:rank1);
newA = newU1*newS1*newV1';
figure(1)
subplot(2,2,1);
pcolor(flip(reshape(A(:,1), [192, 168]))); shading interp; colormap(gray);
title('original face 1 for cropped data');
subplot(2,2,2);
pcolor(flip(reshape(newA(:,1), [192, 168]))); shading interp; colormap(gray);
title('reconstructed face 1 for crooped data');
subplot(2,2,3);
pcolor(flip(reshape(B(:,1), [243,320]))); shading interp; colormap(gray);
title('original face 4 for uncropped data');
subplot(2,2,4);
pcolor(flip(reshape(newB(:,1), [243,320]))); shading interp; colormap(gray);
title('reconstructed face 1 for uncropped data');

%% plot the modes
figure(1)
subplot(2,3,1);
pcolor(flip(reshape(u1(:,1), [192, 168]))); shading interp; colormap(gray);
title('First mode for cropped data');
subplot(2,3,2);
pcolor(flip(reshape(u1(:,2), [192, 168]))); shading interp; colormap(gray);
title('Second mode for crooped data');
subplot(2,3,3);
pcolor(flip(reshape(u1(:,3), [192, 168]))); shading interp; colormap(gray);
title('Third mode for crooped data');
subplot(2,3,4);
pcolor(flip(reshape(u2(:,1), [243,320]))); shading interp; colormap(gray);
title('First mode for uncropped data');
subplot(2,3,5);
pcolor(flip(reshape(u2(:,2), [243,320]))); shading interp; colormap(gray);
title('Second mode for uncropped data');
subplot(2,3,6);
pcolor(flip(reshape(u2(:,3), [243,320]))); shading interp; colormap(gray);
title('Third mode for uncropped data');

%% ---------Part 2-----------------
%% load data.
path = './song/test1';
```

```matlab
% path = './song/test2';
folder = dir(path);

A1 = [];
for i = 3:length(folder)
    filename = folder(i).name;
    [song, Fs1] = audioread(strcat(path, '/', filename)); %load data
    song = song'/ 2;
    song = song(1,:) + song(2,:);    % merge left and right
    % 25 pieces of data per song
    for kk = 40:5:160
        test = song(1, Fs1*kk : Fs1*(kk+5));
%         test = resample(test, 20000, Fs1);
        vector = abs(spectrogram(test));
        vector = reshape(vector, [1, 8*32769]);
        A1 = [A1;vector];
    end
end
A1 = A1';
%%
[u_t1, s_t1, v_t1] = svd(A1 - mean(A1(:)), 'econ');
plot(diag(s_t1) ./ sum(diag(s_t1)), 'ro');
%% 3d plot of the data to see the clusters
vvv = v_t1';
plot3(vvv(2, 1:50), vvv(3, 1:50), vvv(4, 1:50), 'ro'); hold on;
plot3(vvv(2, 51:100), vvv(3, 51:100), vvv(4, 51:100), 'co');
plot3(vvv(2, 101:150), vvv(3, 101:150), vvv(4, 101:150), 'mo'); hold off;
legend('Classic', 'Rap', 'Rock')
xlabel('Mode 2'); ylabel('Mode 3'); zlabel('Mode 4')
%%
acct1_knn = []; acct1_nb = []; acct1_lda = [];
for test_trail = 1:1
    true = [ones(20,1); 2*ones(20,1); 3*ones(20,1)];
    q1 = randperm(50); q2 = randperm(50); q3 = randperm(50);
    xclas = v_t1(1:50, 2:4);
    xpop = v_t1(51:100, 2:4);
    xrock = v_t1(101:150, 2:4);
    xtrain_t1 = [xclas(q1(1:30), :); xpop(q2(1:30), :); xrock(q3(1:30),:)];
    xtest_t1 = [xclas(q1(31:end), :); xpop(q2(31:end), :); xrock(q3(31:end),:)];
    % knn
    ind = knnsearch(xtrain_t1, xtest_t1);
    for i = 1:length(ind)
       if ind(i) ≤ 30
           ind(i) =  1;
       elseif ind(i) ≤ 60
           ind(i) = 2;
       else
           ind(i) = 3;
       end
    end
    temp = [ind==true];
    acct1_knn(test_trail) = sum(temp) / length(temp);
    subplot(3,3,1)
    bar(ind);
    title('kNN');
    xlabel('Test data'); ylabel('Label');

    % naive bayes
    ctrain = [ones(30,1); 2*ones(30,1); 3*ones(30,1)];
    nb = fitcnb(xtrain_t1, ctrain);
    pre = nb.predict(xtest_t1);
    temp = [pre== true];
    acct1_nb(test_trail) = sum(temp) / length(temp);
    subplot(3,3,2);
    bar(pre)
    title('Naive Bayes');
    xlabel('Test data'); ylabel('Label');

    % classify (Built in)
    pre = classify(xtest_t1, xtrain_t1, ctrain);
    temp = [pre== true];
    acct1_lda(test_trail) = sum(temp) / length(temp);
    subplot(3,3,3);
    bar(pre);
    title('LDA');
    xlabel('Test data'); ylabel('Label');
end
acct1_knn = mean(acct1_knn);
acct1_nb = mean(acct1_nb);
acct1_lda = mean(acct1_lda);
result = [acct1_knn;acct1_nb;acct1_lda];

%% ---------Test 3-----------------
%% load data.
path = './song/test3';
folder = dir(path);
A3 = [];
for i = 3:length(folder)
    filename = folder(i).name;
    [song, Fs1] = audioread(strcat(path, '/', filename)); %load data
    song = song'/ 2;
    song = song(1,:) + song(2,:);   % merge left and right
    % 9 pieces of data per song
    for kk = 80:5:120
        test = song(1, Fs1*kk : Fs1*(kk+5));
        vector = abs(spectrogram(test));
        vector = reshape(vector, [1, 8*32769]);
        A3 = [A3;vector];
    end
end
A3 = A3';
%%
[u, s_t3, v_t3] = svd(A3 - mean(A3(:)), 'econ');
plot(diag(s_t3) ./ sum(diag(s_t3)), 'ro');

%%
acct3_knn = []; acct3_nb = []; acct3_lda = [];
for test_trail = 1:100
    true = [ones(24,1); 2*ones(24,1); 3*ones(24,1)];
    q1 = randperm(54); q2 = randperm(54); q3 = randperm(54);
    xclas_t3 = v_t3(1:54, 1:5);
    xpop_t3 = v_t3(55:108, 1:5);
    xrock_t3 = v_t3(109:end, 1:5);
    xtrain_t3 = [xclas_t3(q1(1:30), :); xpop_t3(q2(1:30), :); xrock_t3(q3(1:30),:)];
    xtest_t3 = [xclas_t3(q1(31:end), :); xpop_t3(q2(31:end), :); xrock_t3(q3(31:end),:)];

    % knn
    ind = knnsearch(xtrain_t3, xtest_t3);
    for i = 1:length(ind)
        if ind(i) ≤ 30
            ind(i) =  1;
        elseif ind(i) ≤ 60
            ind(i) = 2;
        else
            ind(i) = 3;
        end
    end
    temp = [ind==true];
    acct3_knn(test_trail) = sum(temp) / length(temp);


    % naive bayes
    ctrain = [ones(30,1); 2*ones(30,1); 3*ones(30,1)];
    nb = fitcnb(xtrain_t3, ctrain);
    pre = nb.predict(xtest_t3);
    temp = [pre==true];
    acct3_nb(test_trail) = sum(temp) / length(temp);

    % classify (Built in)
    pre = classify(xtest_t3, xtrain_t3, ctrain);
    temp = [pre== true];
    acct3_lda(test_trail) = sum(temp) / length(temp);
end
acct3_knn = mean(acct3_knn);
acct3_nb = mean(acct3_nb);
acct3_lda = mean(acct3_lda);
result = [acct3_knn;acct3_nb;acct3_lda];
```