Eigenfaces & Music Genre Identification

Yiling Sun

AMATH 482 HW04 Winter 2019

abstract

This report examines the mechanism of Singular Value Decomposition on dimension-reduction and explores the specific meaning of U,S,V matrices give by SVD. We analyze the influence of the nature of raw data on the performance of SVD. Also, We implement the k nearest-neighbor, Naive Bayes algorithm and linear discriminant Analysis on the data of music sample, to train the classifier to do the music genre identification. The performances of classification algorithms are also compared.

1. Introduction and Overview

This report has two parts:

In the first part, we do the eigenface analysis on the two sets of data, cropped and uncropped, from Yale Face Database. The cropped data is given by the images of only human face, while the uncropped data is given by the images with background. I perform SVD on these data and extract the important information to reconstruct data. By comparing the cropped and uncropped cases, I analyze how does the nature of data influence on the performance of this low-dimension reduction of data. Most importantly, I demonstrate the meaning of the U,S,V matrices from SVD in the case of images data.

In the second part, I train my data source to identify the genre of any given 5 second clip music sample picked from my data. Three test was performed with three different data source. In test 1, I have music samples from Lana Del Rey(Alternative Music), Beattles(Rock Music) and Frederic Chopin(Classic Music), three bands from different genre. For each band, I have 50 5-second samples. In test 2, I have music from Adele, Lady Gaga and Sam Smith. I intentionally picked up their songs in similar pop genre. Still, for each band, I have 50 5-second samples. In test 3, I have three genre, pop, rap and classic. For each genre, I have 150 5-second samples, which is a relatively larger data source. My data source is given by the spectrogram of music after SVD. Then, for each test, I try three supervised classification algorithms to train the data

to identify the given 5-second music samples. To see the performance of each classification method with different kind of data source, I do the cross validation.

2. Theoretical Background

SVD

Single Value decomposition(SVD) is a tool usually carried out when we do the dimension-reduction of potentially low-dimension data. SVD can be applied to all matrices. The basics of SVD is matrix transformation. It basically rotates and stretches, or compresses, a given set of vectors, x, by the matrix A, which often defined as **Ax=b**. For example, in 2 dimensional, the rotation matrix is given by

$$A = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{1}$$

where θ is the angle to rotate. Also, the rotation matrix A is a unitary matrix, $A^{-1} = A^T$. The scaling matrix is given by

$$A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \tag{2}$$

which is a diagonal matrix and α donates the scalar. The combination of the above two matrices control the rotation and scaling in a two-dimensional vector space.

Based on above, Singular value decomposition can factorize matrix into a number of constitutive components all of which have a specific meaning. More concretely, SVD of a function A is

$$A_{m*n} = U_{m*m}\Sigma_{m*n}V_{n*n}, \tag{3}$$

where U and V are orthonormal bases and Σ is diagnal matrix, of which the diagonal entries are nonnegative and ordered from largest to smallest.

Usually for the rectangular matrix. m $\ll$ n, we use the command **[U,S,V] = svd(A,'econ')** that would return as a reduced SVD. In the reduced U, S and V, we get rid of the silence rows in S (all zero), then, only first n columns in U and first n rows of v are calculated, the reduced SVD is in the form:

$$A_{m*n} = U_{m*n}\Sigma_{n*n}V_{n*n}, \tag{4}$$

K Nearest-Neighbors Algorithm

K neaerest-neighbors is a supervised classification algorithm. The label of the new point is determined by the label of k nearest neighbors of the new point. If k = 1, the label of the new point is the same as the nearest point. If k $\geq$, the label of the new point is the majority label among the k points. In Matlab, we apply it by the command **knnsearch**.

Naive Bayes Algorithm

Naive Bayes Algorithm is a supervised classification algorithm. The method is based upon Bayess theorem and the computation of conditional probabilities. Thus one can estimate the label of a new data point based on the prior probability distributions of the labeled data. Hence, we need to properly label data beforehand. In Matlab, we train data by the command **nb = fitcnb(xtrain,ctrain)**,where xtrain is our training data with corresponding label ctrain. Then, by **pre = nb.predict(xtest)**,where xtest is the new point we want to predict. Then, pre would return the prediction.[1]

Linear Discriminant Analysis

Linear Discriminant Analysis is one of the principle technique for the supervised classification. This algorithm look for a linear combination of features to separate two or more classes of objects by projecting it to another space.[2] Still, we need to properly label data beforehand.In Matlab, we can make the prediction by the built in command **classify**.

3. Algorithm Implementation and Development

Eigenfaces Analysis

Prepare Data

Two files of data are given. For the Cropped Data, we have a large file containing 39 under it, in each have some pictures of human face in pgm format. I build a **forloop** to go through 39 files one by one and use **imread** to read in the pictures as matrices. Then, I reshape data of each pictures in column. Thus, I get a big matrix, in which each column keeps the data for one face. For the Cropped data, since all pictures are in one file, I go through this file and read in data in the same way as in Cropped. Then, I change these two matrices to double to allow later modification.

[1] Description from Kutz$_n otes$
[2] Description from Kutz$_n otes$

Eigenfaces Analysis

To do SVD, I minus the the mean of data to normalize it. I apply SVD with 'econ' to get the reduced U, S and V matices. In matrix U, each column contains a orthonormal mode, representing one eigenface. The diagonal entries in matrix S gives the energy of each eigenface. The larger energy is, the more dominant the eigence face is, which means the more feature of the original data can be captured in this mode. Each row of matrix V contains information regarding the each images projection onto the eigenfaces. Since SVD gives us the matrix S in non-increasing order, we can easily recognize the most important modes. By calculating the sum of the energy, we can determine how many modes we need to reconstruct the original data set depending on how much energy we want to capture.

Music Classification

For our three tests, we apply same strategy with different data sources.

Prepare Data

Give music in the file, I use command **audioread** to read in each music and the number of data points for each second. Because the music data contains two columns for left and right channal, I sum them up to be data of the song. In order to keep the data size more manageable, I resample my data with one second 20,000 points. Then, I choose the first 125 second of each song to cut them in to 25 5-second data sample. I use command **spectrogram** to get the spectrogram of each song, because the frequency signature of music are more representative. In the first two test, I have two songs for each class. So, in total, I have 50 sample data for each class, 150 for all. For the 3rd test, I have 6 songs for each genre, so I have 450 sample in total. Then, I apply SVD on my data set to standardize it. Still I minus the mean of data set to normalize it before SVD. With the U, S, V matrices, each row of the matrix V represents each sample in the new base given by matrix U, while each column in U represents a mode in the orthogonal base. In matrix V, each entry represents the feature of each 5 second sample in the corresponding mode. I select the information in 2nd, 3rd, 4rd mode to represent my sample data. Because according to the plotting result, data in different classes are pretty separate that convenience the latter classification.

classification algorithm

I try k nearest-neighbors algorithm, Naive Bayes algorithm and linear discriminant algorithm. I choose 60% of data to be the training set and 30% to be test. My label for the samples are 1,2,3 for each class for convenience. For each algorithm, I calculate the accuracy of the prediction by dividing the correct prediction

by total number of test. For the cross validation, I use the command **randperm** to randomly choose training set and test set from my sample.

K nearest-neighbors algorithm

I choose K = 3, so for the test data, I pick three nearest training data around it to determine its label. Once I get the index of these three points by **knnsearch**, I locate them in my training data to get its label and find the label of majority as my prediction for the test points.

Naive Bayes

To implement Naive Bayes, I simply Matlab code in theoretical background. The return pre is the prediction for the test data.

Linear Discriminant Analysis

With one line code, **pre = classify(xtest,xtrain,ctrain)**. The return pre is the prediction for the test data.

4. Computational Results

Eigenface Analysis

Figure 1 and 2 shows the singular value spectrum of first 50 modes for both cases. In the case of cropped data, there are two prominent modes with energy around 0.055, 4 modes with energy between 0.01 and 0.02 and all other under 0.01, which is pretty small. In the uncropped case, the largest mode is with energy

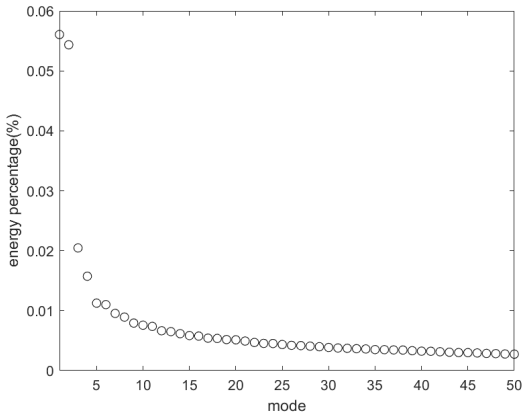

Figure 1: singular value spectrum of first 50 modes for cropped data

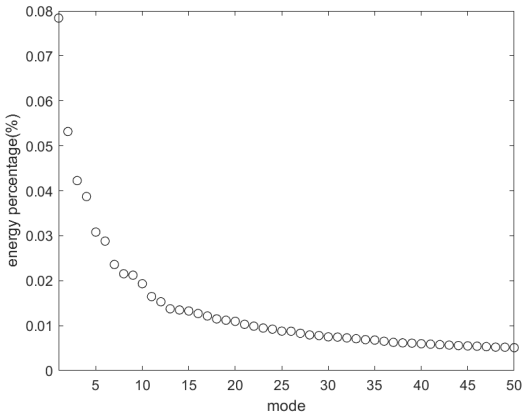

Figure 2: singular value spectrum of first 50 modes uncropped data

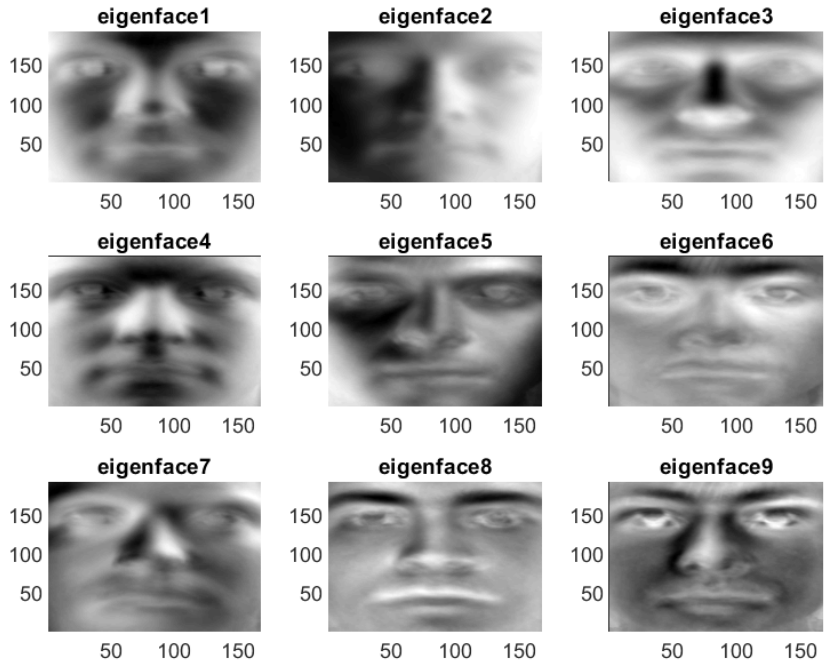

Figure 3: first nine eigenfaces for cropped data

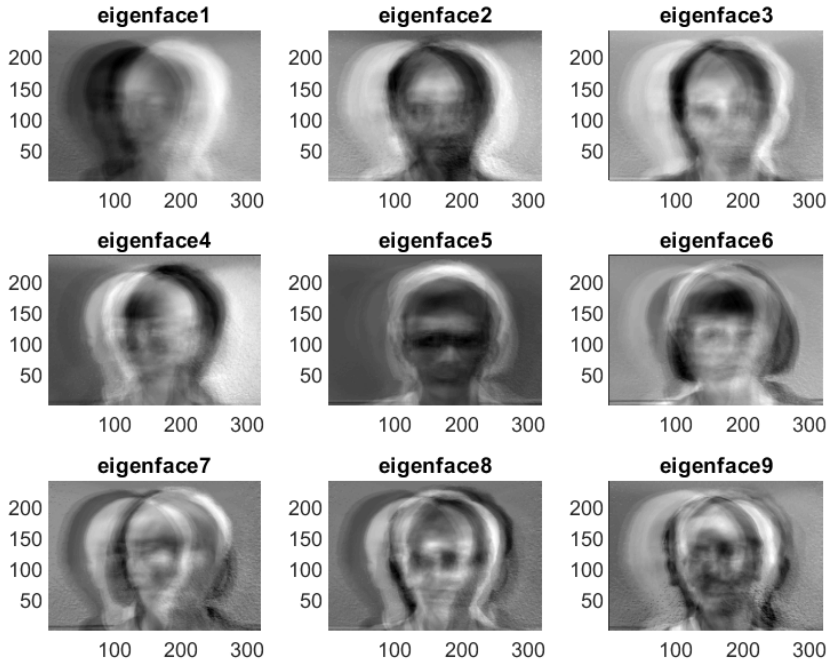

Figure 4: first nine eigenfaces for uncropped data

around 0.08, which is greater than that in cropped case, but there are 8 modes between 0.06 and 0.02. Compare the curve of the energy, we observe that the curve for uncropped data converge slowly to zero than cropped data, which should be caused by the noisy data from the background.

Figure 3 and 4 show the first 9 important eigenface for each case. In cropped case, the information in images are relatively concentrated and even in the first eigenface, it still has the basic face feature. While in uncropped case, since the images are not even aligned, the first eigenface basically shade, containing not much information.

Then, I assume that to capture 90% of the total energy is enough for us to reconstruct the original images set. By calculating, for the cropped data, **1183** modes are needed(rank 1183 for the new face space), while for the uncropped data, **105** is needed(rank 105 for the new face space).

In Figure 5, The upper two images are the first image from the original cropped data set and the first image from the new cropped data set after SVD reconstruction. And the lower twos are from the uncropped. For both cases, we only capture 90% energy of the original data, the images both seems to be lighter than the original one, while the uncropped data, with the noisy information in background, the reconstructed images seems more blurring.

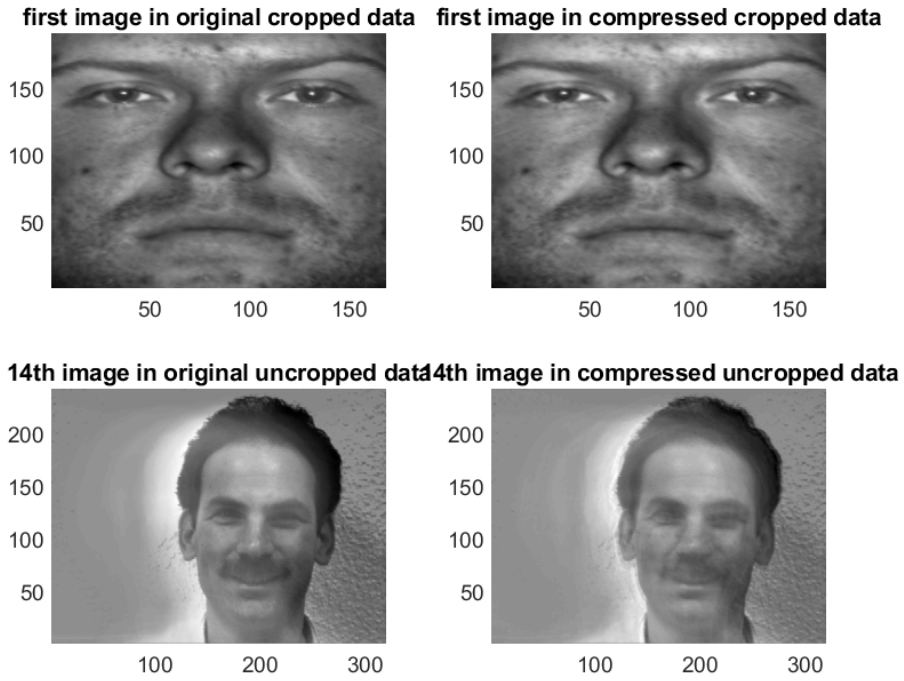

Figure 5: Two images from original images set and the reconstruct images set for the cropped data(upper two) and uncropped data(lower two)

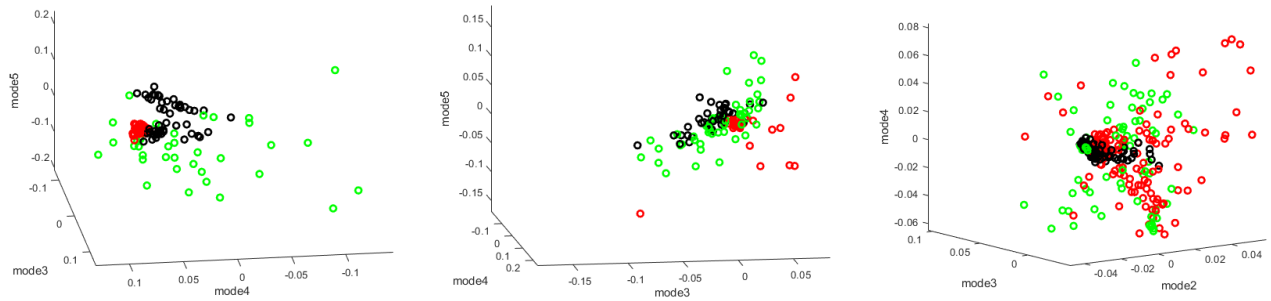

Figure 6: data distribution of test 1, test 2 and test 3(from left to right)

Music Genre Classification

Because our source data is extracting from the feature space given by SVD, I need to determine what modes, in feature space, containing the information given the better clustering data. After I tried for several times, I found that the 2D information give by 2 modes failed. Then, I tried different combination of 3 modes, providing 3D plot. I choose the modes3, 4, 5 for the test 1, modes 3,45 for test 2 and 2,3,4 for test 3. Figure 6,7,8 show the distribution of music sample in these modes.

test 1

Figure 7 shows the result for four times of cross validation. Since we already know that the first 20 is 1, 20 40 is 2 and 40 60 is 3, we can easily observe the performance of each algorithm in each run.
In order to get a more precise result for the performance of each algorithm, I did 1000 times of cross validation and take the average value and its variance, my result is

	mean	variance
knn	0.9152	0.0010
naive bayes	0.9285	6.2636e-04
LDA	0.7272	0.0029

The variance is small, so I assum the result is dependable. In general, knn and naive bayes give accuracy

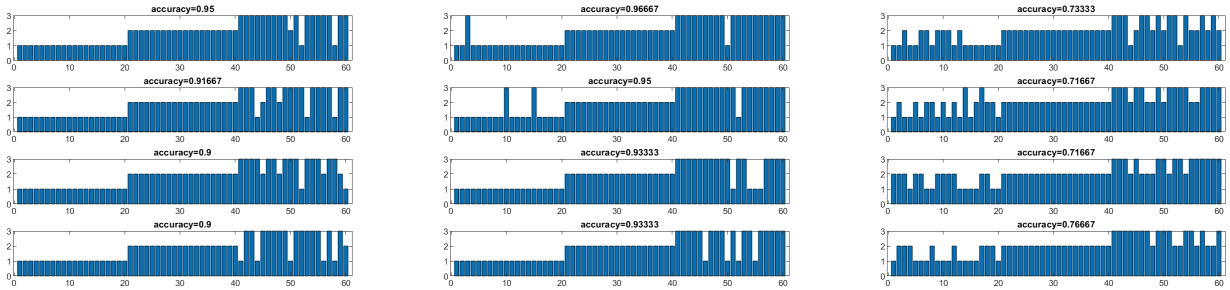

Figure 7: result of four cross validation for knn, naive bayes and LDA for test 1

around 90%, while only 72% for the LDA.

test 2

My result from 1000 times of cross validation is

	mean	variance
knn	0.7364	0.0025
naive bayes	0.5402	0.0024
LDA	0.5054	0.0031

In general, the accuracy is much less than that in the test 1, especially the accuracy of naive bayes drop 35%, from 90% to 54%. LDA is still bad, while knn give a relatively better prediction.

test 3

My result from 1000 times of cross validation is

	mean	variance
knn	0.7891	6.7605e-04
naive bayes	0.6877	8.6791e-04
LDA	0.5922	7.8021e-04

Still, knn had the greatest performance, while LDA did worst.

5. Summary and Conclusions

For part 1, we can conclude that with less noise in images data, the SVD can give better reconstruction result with less information. For the U,S,V matrix given by SVD, U is the new orthonormal bases, in the case of images, eigenfaces, S represents the energy of each bases, V is the information regarding to the images' projection on the eigenfaces. Hence, the rows of V can represent the original data and each entry in row represents its location in that dimension. This strategy is super helpful when we want to do the data classification. Because now we can use less data point to represent one data sample.

In part two, we observe that the classification within more distinct data would give better. Also, as data in one class become more complex, the accuracy of classification goes down. The performance of K nearest-neighbor is generally better than the other two. Naive bayes has a bad performance at differentiating similar classes, while the linear discriminant analysis are always bad, which may caused by the mixing of some data points.

6. Appendix A MATLAB functions used and brief implementation explanation

[y,Fs] = audioread(filename) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.[3]

new = resample(tsin,timevec,interpmethod) resamples tsin using a specified interpolation method.[4]

p = randperm(n) returns a row vector containing a random permutation of the integers from 1 to n inclusive.[5]

Y = double(X) converts the values in X to double precision.[6]

[U,S,V] = svd(A,'econ') [U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'. Removing these zeros and columns can improve execution time and reduce storage requirements without compromising the accuracy of the decomposition.[7]

M = mean(A) returns the mean of the elements in A.[8]

M = var(A) returns the variance of the elements in A.[9]

x = diag(A) returns a column vector of the main diagonal elements of A.[10]

S = sum(A) returns the sum of the elements of A. [11]

Idx = knnsearch(X,Y,Name,Value) returns Idx with additional options specified using one or more name-value pair arguments. For example, you can specify the number of nearest neighbors to search for and the distance metric used in the search.[12]

Mdl = fitcnb(Tbl,formula) returns a multiclass naive Bayes model (Mdl), trained by the predictors in table Tbl.[13]

class = classify(sample,training,group) classifies each row of the data in sample into one of the groups in training. sample and training must be matrices with the same number of columns. group is a grouping variable for training. Its unique values define groups; each element defines the group to which the corresponding row of training belongs.[14]

[3]Description from MathWorks
[4]Description from MathWorks
[5]Description from MathWorks
[6]Description from MathWorks
[7]Description from MathWorks
[8]Description from MathWorks
[9]Description from MathWorks
[10]Description from MathWorks
[11]Description from MathWorks
[12]Description from MathWorks
[13]Description from MathWorks
[14]Description from MathWorks

7. Appendix B MATLAB codes

Part 1

```
clear all; close all; clc;
%%
M=[];
for i = 1:39
    if i<10
        file_name = strcat('CroppedYale/yaleB0',num2str(i),'/');
    else
        file_name = strcat('CroppedYale/yaleB',num2str(i),'/');
    end
    file = dir(file_name);

    for j = 1:length(file)
        if file(j).bytes>0
            data = imread(strcat(file_name,'/',file(j).name));
            data_col = reshape(data,[],1);
            M = [M data_col];
        end
    end
end
M=double(M);
%%
% [m,n]=size(M); % compute data size
% mn=mean(M,2); % compute mean for each row
% M=M-repmat(mn,1,n); % subtract mean

[u1,s1,v1]=svd(M-mean(M(:)),'econ'); % perform the SVD
%%
% plot(diag(s),'ko','Linewidth',[2]);
% set(gca,'Xlim',[0,50]);
energy1=diag(s1)/sum(diag(s1));
figure()
plot(energy1,'ko')
set(gca,'Xlim',[1,50])
xlabel('mode'); ylabel('energy percentage(%)')
```

```
saveas(gcf,'mode1.png')
%%
figure()
for i=1:9
    subplot(3,3,i)
    face1 = reshape(u1(:,i),192,168);
    pcolor(flipud(face1)), shading interp, colormap(gray);
    title(['eigenface',num2str(i)])
end
saveas(gcf,'eigenface1.png')
%%
for j = 1:2400
    if sum(energy1(1:j))>0.9
        break
    end
end
%%
u1_new = u1(:,1:j);
s1_new = s1(1:j,1:j);
v1_new = v1(:,1:j);

new1 = u1_new*s1_new*v1_new';


%% Uncropped
path = 'UncroppedYale/yalefaces/';
file = dir(path);
M2=[];
for i = 1:length(file)
    if file(i).bytes > 0
        filename = file(i).name;
        data = imread(strcat(path,filename));
        data_col = reshape(data,[],1);
        M2 = [M2 data_col];
    end
end
M2 = double(M2);
%%
[m,n]=size(M2); % compute data size
mn=mean(M2,2); % compute mean for each row
M2=M2-repmat(mn,1,n); % subtract mean

```

```
[u2,s2,v2]=svd(M2-mean(M2(:)),'econ'); % perform the SVD
% plot(diag(s),'ko','Linewidth',[2]);
% set(gca,'Xlim',[0,50]);
energy2=diag(s2)/sum(diag(s2));
%%
figure()
plot(energy2,'ko')
set(gca,'Xlim',[1,50])
xlabel('mode'); ylabel('energy percentage(%)')
saveas(gcf,'mode2.png')
%%
figure()
for i=1:9
    subplot(3,3,i)
    face1 = reshape(u2(:,i),243,320);
    pcolor(flipud(face1)), shading interp, colormap(gray);
    title(['eigenface',num2str(i)])
end
saveas(gcf,'eigenface2.png')
%%
for rank2 = 1:2400
    if sum(energy2(1:rank2))>0.9
        break
    end
end
%%
u2_new = u2(:,1:rank2);
s2_new = s2(1:rank2,1:rank2);
v2_new = v2(:,1:rank2);

new2 = u2_new*s2_new*v2_new';
%%
figure()
subplot(2,2,1)
face_old1=reshape(M(:,1),192,168); % pull out modes
pcolor(flipud(face_old1)), shading interp, colormap(gray);
title('first image in original cropped data')
subplot(2,2,2)
face_new1=reshape(new1(:,1),192,168); % pull out modes
pcolor(flipud(face_new1)), shading interp, colormap(gray);
title('first image in compressed cropped data')
subplot(2,2,3)
```

```
face_old2=reshape(M2(:,14),243,320); % pull out modes
pcolor(flipud(face_old2)), shading interp, colormap(gray);
title('14th image in original uncropped data')
subplot(2,2,4)
face_new2=reshape(new2(:,14),243,320); % pull out modes
pcolor(flipud(face_new2)), shading interp, colormap(gray);
title('14th image in compressed uncropped data')
saveas(gcf,'result.png')
```

test 1

```
clear all; close all; clc;
%%
path = './music1/';
file = dir(path);
data = [];
for i = 3:8
    filename = file(i).name;
    [song, Fs] = audioread(strcat(path,'/',filename));
    song = song(:,1)+song(:,2);
    song = resample(song,20000,Fs);
    Fs=20000;
    parts = [];
    for j = 1:5:125
        five = song(Fs*j:Fs*(j+5),1);
        five_spec = abs(spectrogram(five));
        five_spec = reshape(five_spec,1,16385*8);
        parts = [parts five_spec'];
    end
    data = [data parts];
end
%%

[u,s,v]=svd(data-mean(data(:)),'econ');

plot(diag(s),'ko','Linewidth',[2])
%%
plot3(v(1:50,3),v(1:50,4),v(1:50,5),'ko','Linewidth',[2]),hold on
plot3(v(51:100,3),v(51:100,4),v(51:100,5),'ro','Linewidth',[2]);
plot3(v(101:150,3),v(101:150,4),v(101:150,5),'go','Linewidth',[2]);
```

```
xlabel('mode3'),ylabel('mode4'),zlabel('mode5'),

%%
knn_result = [];
for j = 1:1000
    q1 = randperm(50);
    q2 = randperm(50);
    q3 = randperm(50);

    xbtl = v(1:50,3:5);
    xcpn = v(51:100,3:5);
    xlana = v(101:end,3:5);
    xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
    xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
    ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
    ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];

    index = knnsearch(xtrain, xtest,'K',3);
    correct = 0;
    prediction = [];
    for i = 1:length(index)
        knn3 = index(i,:);
        int_pre = [ctrain(knn3(1)) ctrain(knn3(2)) ctrain(knn3(3))];
        pre = mode(int_pre);
        prediction = [prediction pre];
        if pre == ctest(i,1)
            correct = correct+1;
        end
    end
%     subplot(4,1,j);
%     bar(prediction);
    accuracy = correct/length(index);
    knn_result = [knn_result accuracy];
%     title(['accuracy=',num2str(accuracy)])
end
ave_knn_result = mean(knn_result)
var_knn_result = var(knn_result)
% saveas(gcf,'knn1.png')
%% Naive Bayes supreviese
nb_result = [];
for j = 1:1000
    q1 = randperm(50);
```

```
72      q2 = randperm(50);
73      q3 = randperm(50);
74
75      xbtl = v(1:50,3:5);
76      xcpn = v(51:100,3:5);
77      xlana = v(101:end,3:5);
78      xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
79      xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
80      ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
81      ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];
82
83      nb = fitcnb(xtrain,ctrain);
84      pre = nb.predict(xtest);
85
86      correct = 0;
87      times = size(xtest,1);
88      for i = 1:times
89          if pre(i,1) == ctest(i,1)
90              correct = correct+1;
91          end
92      end
93      accuracy = correct/times;
94      nb_result = [nb_result accuracy];
95  %     subplot(4,1,j);
96  %     bar(pre);
97  %     title(['accuracy=',num2str(accuracy)])
98  end
99  ave_nb_result = mean(nb_result)
100 var_nb_result = var(nb_result)
101 % saveas(gcf,'nb1.png')
102 %% LDA supervise
103 lda_result = [];
104 for j = 1:1000
105     q1 = randperm(50);
106     q2 = randperm(50);
107     q3 = randperm(50);
108
109     xbtl = v(1:50,3:5);
110     xcpn = v(51:100,3:5);
111     xlana = v(101:end,3:5);
112     xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
113     xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
```

```
    ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
    ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];

    pre = classify(xtest,xtrain,ctrain);

    correct = 0;
    times = size(xtest,1);
    for i = 1:times
        if pre(i,1) == ctest(i,1)
            correct = correct+1;
        end
    end
    accuracy = correct/times;
    lda_result = [lda_result accuracy];
%     subplot(4,1,j);
%     bar(pre);
%     title(['accuracy=',num2str(accuracy)])
end
% saveas(gcf,'lda1.png')
ave_lda_result = mean(lda_result)
var_lda_result = var(lda_result)
```

test 2

```
clear all; close all; clc;
%%
path = './music2/';
file = dir(path);
data = [];
for i = 3:8
    filename = file(i).name;
    [song, Fs] = audioread(strcat(path,'/',filename));
    song = song(:,1)+song(:,2);
    song = resample(song,20000,Fs);
    Fs=20000;
    parts = [];
    for j = 1:5:125
        five = song(Fs*j:Fs*(j+5),1);
        five_spec = abs(spectrogram(five));
        five_spec = reshape(five_spec,1,16385*8);
```

```
        parts = [parts five_spec'];
    end
    data = [data parts];
end
%%

[u,s,v]=svd(data-mean(data(:)),'econ');

% plot(diag(s),'ko','Linewidth',[2])

%%
plot3(v(1:50,3),v(1:50,4),v(1:50,5),'ko','Linewidth',[2]),hold on
plot3(v(51:100,3),v(51:100,4),v(51:100,5),'ro','Linewidth',[2]);
plot3(v(101:150,3),v(101:150,4),v(101:150,5),'go','Linewidth',[2]);
xlabel('mode3'),ylabel('mode4'),zlabel('mode5'),

%%
knn_result = [];
for j = 1:1000
    q1 = randperm(50);
    q2 = randperm(50);
    q3 = randperm(50);

    xbtl = v(1:50,3:5);
    xcpn = v(51:100,3:5);
    xlana = v(101:end,3:5);
    xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
    xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
    ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
    ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];

    index = knnsearch(xtrain, xtest,'K',3);
    correct = 0;
    for i = 1:length(index)
        knn3 = index(i,:);
        int_pre = [ctrain(knn3(1)) ctrain(knn3(2)) ctrain(knn3(3))];
        pre = mode(int_pre);
        if pre == ctest(i,1)
            correct = correct+1;
        end
    end
    accuracy = correct/length(index);
```

```
59      knn_result = [knn_result accuracy];
60  end
61  ave_knn_result = mean(knn_result)
62  var_knn_result = var(knn_result)
63  %% Naive Bayes supreviese
64  nb_result = [];
65  for j = 1:1000
66      q1 = randperm(50);
67      q2 = randperm(50);
68      q3 = randperm(50);
69
70      xbtl = v(1:50,3:5);
71      xcpn = v(51:100,3:5);
72      xlana = v(101:end,3:5);
73      xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
74      xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
75      ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
76      ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];
77
78      nb = fitcnb(xtrain,ctrain);
79      pre = nb.predict(xtest);
80
81      correct = 0;
82      times = size(xtest,1);
83      for i = 1:times
84          if pre(i,1) == ctest(i,1)
85              correct = correct+1;
86          end
87      end
88      accuracy = correct/times;
89      nb_result = [nb_result accuracy];
90  %     subplot(4,1,j);
91  %     bar(pre);
92  end
93  ave_nb_result = mean(nb_result)
94  var_nb_result = var(nb_result)
95  %% LDA supervise
96  lda_result = [];
97  for j = 1:1000
98      q1 = randperm(50);
99      q2 = randperm(50);
100     q3 = randperm(50);
```

```
101
102     xbtl = v(1:50,3:5);
103     xcpn = v(51:100,3:5);
104     xlana = v(101:end,3:5);
105     xtrain = [xbtl(q1(1:30),:); xcpn(q2(1:30),:); xlana(q3(1:30),:)];
106     xtest = [xbtl(q1(31:end),:); xcpn(q2(31:end),:); xlana(q3(31:end),:)];
107     ctrain=[ones(30,1); 2*ones(30,1); 3*ones(30,1)];
108     ctest=[ones(20,1); 2*ones(20,1); 3*ones(20,1)];
109
110     pre = classify(xtest,xtrain,ctrain);
111
112     correct = 0;
113     times = size(xtest,1);
114     for i = 1:times
115         if pre(i,1) == ctest(i,1)
116             correct = correct+1;
117         end
118     end
119     accuracy = correct/times;
120     lda_result = [lda_result accuracy];
121 %     subplot(4,1,j);
122 %     bar(pre);
123 end
124 ave_lda_result = mean(lda_result)
125 var_lda_result = var(lda_result)
```

test 3

```
1  clear all; close all; clc;
2  %%
3  path = './music3/';
4  file = dir(path);
5  data = [];
6  for i = 3:20
7      filename = file(i).name;
8      [song, Fs] = audioread(strcat(path,'/',filename));
9      song = song(:,1)+song(:,2);
10     song = resample(song,20000,Fs);
11     Fs=20000;
12     parts = [];
```

```
    for j = 1:5:125
        five = song(Fs*j:Fs*(j+5),1);
        five_spec = abs(spectrogram(five));
        five_spec = reshape(five_spec,1,16385*8);
        parts = [parts five_spec'];
    end
    data = [data parts];
end

%%
[m,n]=size(data); % compute data size
mn=mean(data,2); % compute mean for each row
data=data-repmat(mn,1,n); % subtract mean
[u,s,v]=svd(data,'econ');

plot(diag(s),'ko','Linewidth',[2])

third = n /3;
plot3(v(1:third,2),v(1:third,3),v(1:third,4),'ko','Linewidth',[2]),hold on
plot3(v((third+1):2*third,2),v((third+1):2*third,3),v((third+1):2*third,4),'ro','Linewidth',[2]);
plot3(v((2*third+1):end,2),v((2*third+1):end,3),v((2*third+1):end,4),'go','Linewidth',[2]);
xlabel('mode2'),ylabel('mode3'),zlabel('mode4'),

%%
knn_result = [];
for j = 1:1000

    q1 = randperm(third);
    q2 = randperm(third);
    q3 = randperm(third);

    xbtl = v(1:third,2:4);
    xcpn = v((third+1):(2*third),2:4);
    xlana = v((2*third+1):end,2:4);
    num_train = third*(3/5);
    num_test = third*(2/5);
    xtrain = [xbtl(q1(1:num_train),:); xcpn(q2(1:num_train),:); xlana(q3(1:num_train),:)];
    xtest = [xbtl(q1((num_train+1):end),:); xcpn(q2((num_train+1):end),:); ...
        xlana(q3((num_train+1):end),:)];
    ctrain=[ones(num_train,1); 2*ones(num_train,1); 3*ones(num_train,1)];
    ctest=[ones(num_test,1); 2*ones(num_test,1); 3*ones(num_test,1)];

```

```
    index = knnsearch(xtrain, xtest,'K',3);
    correct = 0;
    for i = 1:length(index)
        knn3 = index(i,:);
        int_pre = [ctrain(knn3(1)) ctrain(knn3(2)) ctrain(knn3(3))];
        pre = mode(int_pre);
        if pre == ctest(i,1)
            correct = correct+1;
        end
    end
    accuracy = correct/length(index);
    knn_result = [knn_result accuracy];
end
ave_knn_result = mean(knn_result)
var_knn_result = var(knn_result)
%% Naive Bayes supreviese
nb_result = [];
for j = 1:1000

    q1 = randperm(third);
    q2 = randperm(third);
    q3 = randperm(third);

    xbtl = v(1:third,2:4);
    xcpn = v((third+1):(2*third),2:4);
    xlana = v((2*third+1):end,2:4);
    num_train = third*(3/5);
    num_test = third*(2/5);
    xtrain = [xbtl(q1(1:num_train),:); xcpn(q2(1:num_train),:); xlana(q3(1:num_train),:)];
    xtest = [xbtl(q1((num_train+1):end),:); xcpn(q2((num_train+1):end),:); ...
        xlana(q3((num_train+1):end),:)];
    ctrain=[ones(num_train,1); 2*ones(num_train,1); 3*ones(num_train,1)];
    ctest=[ones(num_test,1); 2*ones(num_test,1); 3*ones(num_test,1)];

    nb = fitcnb(xtrain,ctrain);
    pre = nb.predict(xtest);

    correct = 0;
    times = size(xtest,1);
    for i = 1:times
        if pre(i,1) == ctest(i,1)
            correct = correct+1;
```

```
95          end
96      end
97      accuracy = correct/times;
98      nb_result = [nb_result accuracy];
99  %     subplot(4,1,j);
100 %     bar(pre);
101 end
102 ave_nb_result = mean(nb_result)
103 var_nb_result = var(nb_result)
104 %% LDA supervise
105 lda_result = [];
106 for j = 1:1000
107
108     q1 = randperm(third);
109     q2 = randperm(third);
110     q3 = randperm(third);
111
112     xbtl = v(1:third,2:4);
113     xcpn = v((third+1):(2*third),2:4);
114     xlana = v((2*third+1):end,2:4);
115     num_train = third*(3/5);
116     num_test = third*(2/5);
117     xtrain = [xbtl(q1(1:num_train),:); xcpn(q2(1:num_train),:); xlana(q3(1:num_train),:)];
118     xtest = [xbtl(q1((num_train+1):end),:); xcpn(q2((num_train+1):end),:); ...
            xlana(q3((num_train+1):end),:)];
119     ctrain=[ones(num_train,1); 2*ones(num_train,1); 3*ones(num_train,1)];
120     ctest=[ones(num_test,1); 2*ones(num_test,1); 3*ones(num_test,1)];
121
122     pre = classify(xtest,xtrain,ctrain);
123
124     correct = 0;
125     times = size(xtest,1);
126     for i = 1:times
127         if pre(i,1) == ctest(i,1)
128             correct = correct+1;
129         end
130     end
131     accuracy = correct/times;
132     lda_result = [lda_result accuracy];
133 %     subplot(4,1,j);
134 %     bar(pre);
135 end
```

```
136  ave_lda_result = mean(lda_result)
137  var_lda_result = var(lda_result)
```