Music classification - PCA

Hyesu Lee

March 6, 2020

Abstract

For this report, I will explain the definition of machine learning algorithms and the role of principal component analysis(PCA) in context of machine learning. The report will cover two parts: 1) understanding the PCA using images and 2) performing music classification. The code is implemented in MATLAB.

1 Introduction and Overview

The report will have two parts. First part will cover the understanding of principal component analysis and singular value decomposition regarding image processing. In second part, student is asked to use one of the classification algorithm with five second music samples to classify their genres. In this assignment, I used linear discriminant analysis as statistical learning algorithm for music classification.

2 Theoretical Background

2.1 Machine learning

Machine learning could be define as learning from data. With large data, machine learning(ML) algorithms find some pattern or structure to cluster and classify. There are three areas of machine learning algorithm: unsupervised learning, supervised learning, and reinforcement learning. For the purpose of the report, I will only elaborate on unsupervised and supervised learning.

2.2 Unsupervised learning

Unsupervised learning assumes that the given data set is unknown. The unsupervised learning algorithm finds some patterns within the given data set and make clusters of the data such that any data point within the cluster is similar to each other. The main idea is that the algorithm is to find pattern and the labels for each data set are not needed. Followings are some of the unsupervised learning algorithm: K-means, Mixture model.

2.2.1 K - means

K means algorithm finds K clusters with the given data set by using the idea of center of mass. *K-means* algorithm:

1. Find k random points within the range of data set.
2. Make a group of data points such that any data point belongs to the closest random point.
3. Within the group, find the center of mass and defines those points as new "random points".
4. Repeat the procedures until the random point converges and those groups are the clusters.

2.2.2 Mixture model - Gaussian distribution

Mixture model algorithm assumes that the given data is created by data from different data sets. Therefore, the distribution of the given data set could be represented as a sum of statistical distribution. In the equation 1, K is the number of statistical distribution and P represents corresponding distribution. Thus Mixture model algorithm finds specific distributions in the given set and outputs each distributions as a cluster. The most common model is Gaussian mixture model, which assumes that every distribution is a Gaussian distribution.

$$P(x|\theta) = \sum_{K=1}^{K} \alpha_K P(x, \theta_K) \tag{1}$$

2.3 Supervised learning

Supervised learning assume that given data is known. In other words, the data set has been already clustered and labels for each data set are given . Supervised learning algorithms "train" the machine with the given data set such that when a new data is given, the machine could classify the new data. Therefore, supervised learning algorithm contains testing set, training set, and label set. The label set represents the cluster each data point belongs. Lastly, it is important to cross-validate to prevent one specific discrimination based of one specific training set. Cross-validate could be done by repeating the training and testing process with the newly selected data points.

2.3.1 K-nearest-neighbors

K nearest neighbors (KNN) algorithm classify the given data point based on the K nearest values. Suppose there are 3 nearest values. If the two nearest values have label of dog and the other nearest value has label of cat, then the given data point is classified as dog. Also, KNN is capable of generating non-linear classification line.

2.3.2 Naive Bayes

Naive Bayes algorithm calculates the probability of each classes given some data. Equation 2 is representation of Naive Bayes algorithm with two classes. The right side of equation 2 is simpler to compute than the left side. It is how the Naive Bayes algorithm classify a new given data point.

$$\frac{P(1|x)}{P(0|x)} = \frac{P(x|1) * P(1)}{P(x|0) * P(0)} \tag{2}$$

2.3.3 Linear discriminant analysis

Linear discriminant analysis draws a line that separates the clusters. In other words, it projects every data points onto some basis and find a linear line for the classification. This algorithm is powerful to use with PCA, because PCA allows to find new bases such that when the data points projected onto those new bases, the clusters get separated nicely. Then, one could use linear discriminant analysis to find a linear classification line. Once the line is defined, then the any new data point could be classified easily.

2.3.4 Support vector machine

Support vector machine(SVM) finds the maximal separation line of given training set. Similar to linear discriminate analysis, SVM finds a line for separation, but SVM optimize the thickness of the line such that it has the maximal separation. The points on the edge of the cluster are called support vectors and SVM finds a line based on those support vectors. Any points on the wrong side would be a penalized factor. SVM allows to have curved surfaces in the higher spaces. Lastly, SVM does not require huge data set, because the only important points are the support vectors.

2.3.5 Classification and regression tree(CART)

Classification and regression tree(CART) generates a binary tree that breaks down the data set with some sorting based on the pattern of each cluster. By some binary separation, the new data could be classified as one of the cluster.

3 Algorithm Implementation and Development

3.1 Part 1

There are two different data sets given: Cropped images and original images. The purpose of part 1 is to interpret the singular value decomposition with a matrix of multiple pictures and how each principal component captures the differences of the images. Following implementation is for both cropped and uncropped data set analysis

1. Load the data set and reshape each images as a vector.
2. Combine all the image vectors to make a matrix that each column represents each image.
3. Perform singular value decomposition and plot the singular values
4. Plot the principal components correspond to the important singular values.
5. Plot the feature vectors, V.

3.2 Part 2

In part 2, I used five seconds music clips to perform music genre or band classification. For each of three tests, the sample clips are different: different band of different genre, different bands of same genre, and various band for each genre. For the supervised learning algorithm, I used linear discriminant analysis. For each music cluster, there are 90 sample clips and I portioned 70 sample clips as training set and 20 sample clips as testing sets. Following implementation is for all three tests.

1. Load and parse the music clips to same time length.
2. Perform Fourier transform and create spectrogram on each music clip.
3. Reshape each music clip to be a vector and make a matrix for each cluster.
4. Randomly choose training set and test set from each cluster.
5. Perform SVD on training set and find the significant singular values.
6. Project the cluster matrices onto the reduced principal components.
7. By using the projected features, use linear discriminant analysis to train and test.
8. Find the success rate of the classification.
9. Repeat procedure 6 8 with newly selected train set and test set for cross validation.
10. Find success rate distribution of all the cross validations.

4 Computational Results

4.1 Part 1

As a output of SVD, there are three matrices: U, Σ, V. U is a matrix of principal components such that each vector represent a new basis for the given data. Σ is a diagonal matrix of singular value. Singular value could be used to compute the energy that each principal component captures. Lastly, U is a matrix and each vector captures the features of the given data.
When SVD is performed on cropped images, the first four singular values captures most of the energy in cropped images, Figure 1. The first principal components, Figure 3, captures the basic features of the face, eyes, noise,and mouth. Then, the second principal component,Figure 4, captures the shades on the face. The figure 5 and figure 6 shows that features in each face could be described by adding or subtracting the values on principal components.
When SVD is performed on cropped face, the first two singular values captures the most of energy of the given data. The first principal component, Figure 7, captures the head silhouette and little bit of hair line and facial hair. On the second principal component, Figure 8, captures the placement of the head.
According to the principal component representation, the first couple principal components of the cropped images captures specific features while principal components of uncropped images, capture general silhouette.

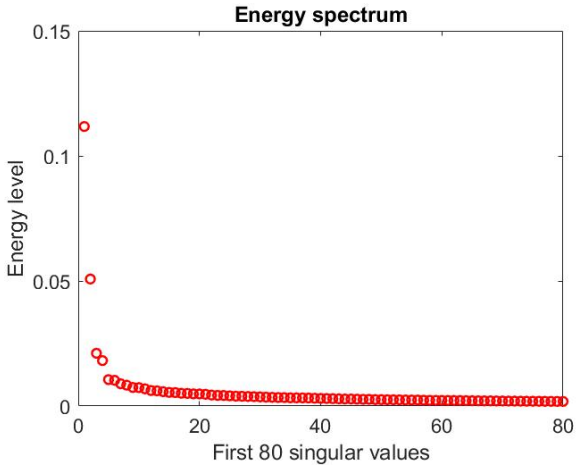


Figure 1: Cropped face singular value energy spectrum

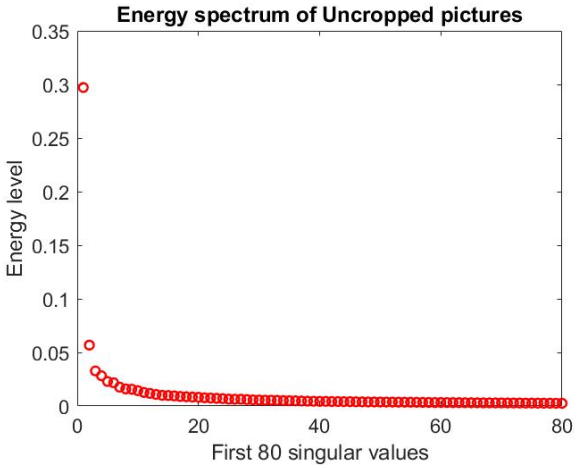


Figure 2: Uncropped face singular value energy spectrum

4.2 Part 2

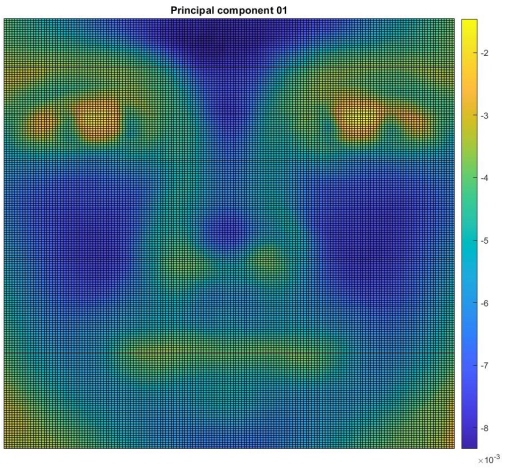
As mentioned before, I used linear discriminant analysis for the classification algorithm. For the comparison, each test case will have different numbers of principal components such that total energy captured by is 80% of total energy. I cross validate each test 1000 times and plot the distribution.

4.2.1 test 1: Classify genre with one musician per each genre

As shown in Figure 9, 59 singular values captures 80.85%. The distribution of success rate, 10, has mean of 82.9083% with standard deviation of 3.9853%.

4.2.2 test 2: Classify musicians who are in same genre

As shown in Figure 11, 76 singular values captures 80.67%. The distribution of success rate, 12, has mean of 64.995% with standard deviation of 4.4872%.



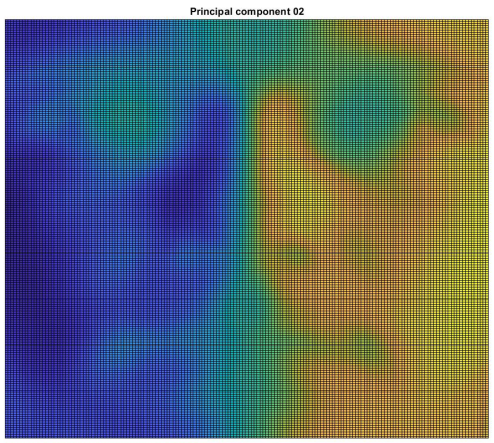
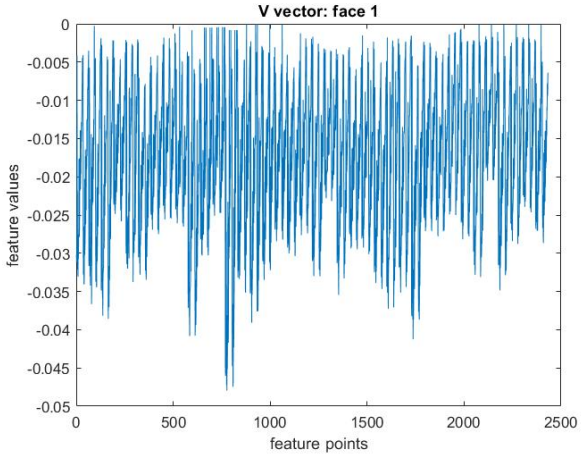


Figure 3: Cropped: Representation of the first principal component

Figure 4: Cropped: Representation of the second principal component



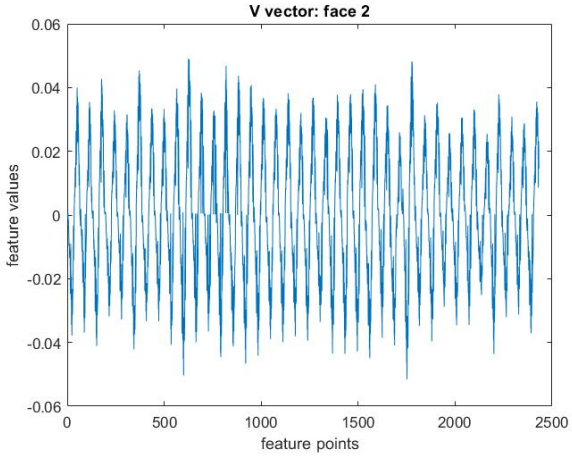


Figure 5: Features of cropped face 01 on first principal component

Figure 6: Features of cropped face 02 on first principal component

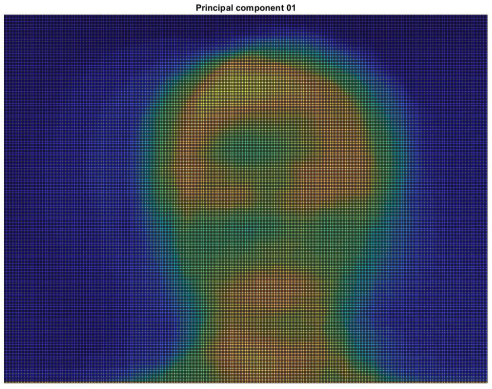
4.2.3 test 3: Classify genre with multiple musicians

As shown in Figure 13, 70 singular values captures 80.41%. The distribution of success rate, 14, has mean of 80.325% with standard deviation of 4%.

5 Summary and Conclusions

5.1 Part 1

The principal component representation captures more detail if the data is parsed nicely. Also, the V matrix achieve the features of the each image by add or subtract the values in principal components.



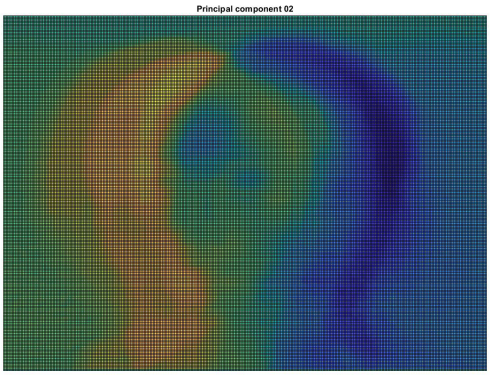
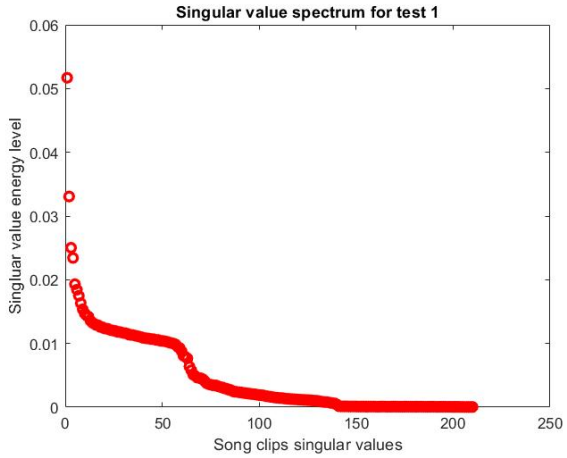


Figure 7: Uncropped: Representation of the first principal component

Figure 8: Uncropped: Representation of the second principal component



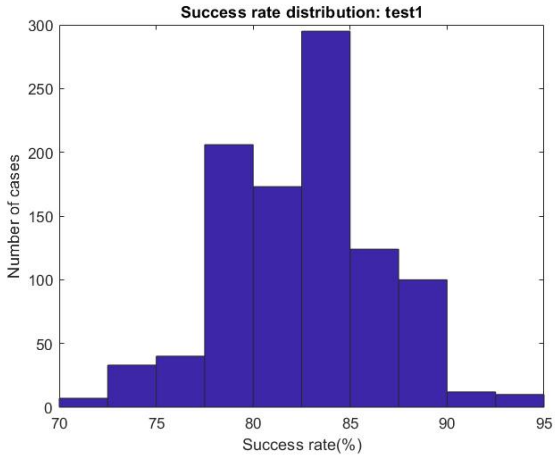


Figure 9: test1:singular value energy spectrum

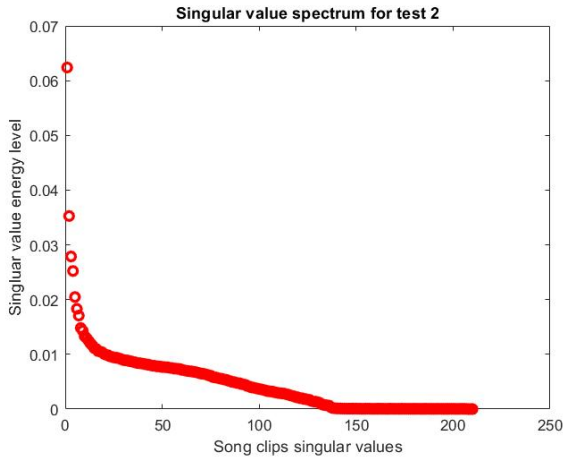
Figure 10: test1:success rate distribution

5.2 Part 2

The mean success rates of test 1 and test 3 are similar, 80%, but the mean success rate of test 2 is lower, 65%. The difference between those two groups are that test 1 and test 3 sample clips are extracted from different genres, while test 2 sample clips are extracted from same genre. Features in test 2 sample clips are similar to each other such that there is less clear separation compare to test 1 and test 3. Therefore, the success rate in test 2 is lower than the success rate of test 1 and 3.

Appendix A MATLAB Functions

- `B = reshape(A, sz)` return reshaped matrix `A`, using the size vector `sz` for size of matrix `B`.
- `imshow(A)` display the image A.
- `[U,S,V] = svd(X, 'econ')` produces a diagonal matrix S and unitary matrices U and V so that $X = U * S * V'$. With 'econ' parameter, executes the reduced SVD.



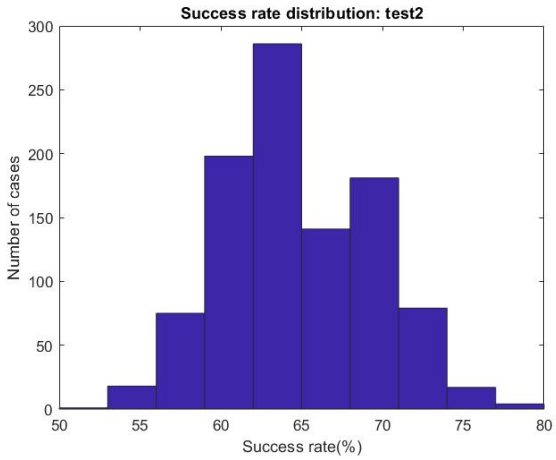
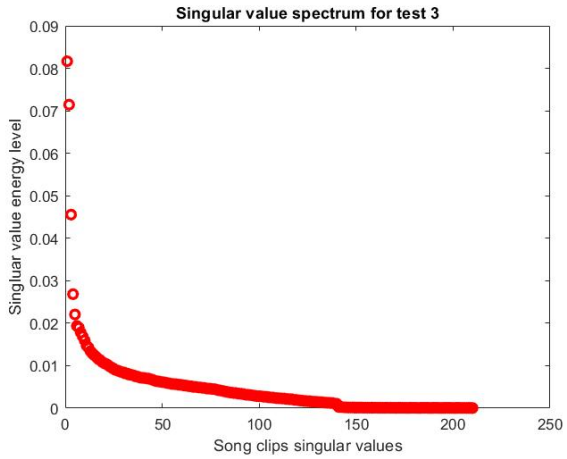


Figure 11: test2:singular value energy spectrum

Figure 12: test2:success rate distribution



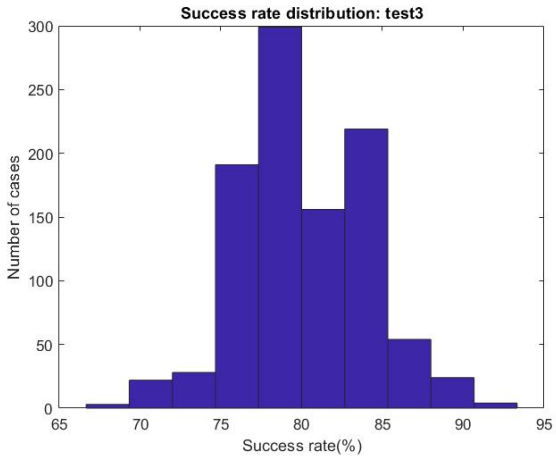


Figure 13: test3:singular value energy spectrum

Figure 14: test3:success rate distribution

- `lambda = diag(S)` returns the diagonal entries of matrix S as vector.
- `P = randperm(N)` returns a vector containing a random permutation of the integers 1:N.
- `Classified = classify(sample, training, group)` classifies each row of the data in sample into one of the groups in training.
- `num = nnz(S)` returns the number of nonzero elements in S.
- `T = num2str(X)` converts the matrix X into its character representation T.
- `PD = fitdist(X,distname)` fits the probability distribution distname to the data in the column vector X, and returns an object PD representing the fitted distribution.
- `N = floor(X)` rounds the elements of X to the nearest integers towards minus infinity.
- `y = fftshift(x)` return the rearranged Fourier transform x which contains zero-frequency component at the center of the array.

- `y = fft(x)` returns the discrete Fourier transform of vector X.

Appendix B Github page

https://github.com/HyesLee99/Data-analysis/tree/master/HW4

Appendix C MATLAB Code

```
clear all; clc;

mainFolder1 = dir('yalefaces_cropped/CroppedYale');
mainFolder1 = mainFolder1(3:length(mainFolder1));
CROP = [];
ave_face = [];
for i=1:length(mainFolder1)
    name = mainFolder1(i).name;
    a = ['yalefaces_cropped/CroppedYale', '/', name];
    subfolder=dir(a);
    subfolder = subfolder(3:end);
    Cropped = [];
    for j = 1:length(subfolder)
        b = [a, '/', subfolder(j).name];
        data = imread(b);
        data = reshape(data,192*168,1);
        Cropped = [Cropped data];
        CROP = [CROP data];
    end
    ave_face = [ave_face sum(Cropped,2)/length(Cropped(1,:))];
end
%%
save CROP.dat CROP -ascii;
save ave_face.dat ave_face -ascii;
%% bring data
clear all; clc; close all;
load CROP.dat;
load ave_face.dat;
%% SVD
CROP = double(CROP);
[U,S,V] = svd(CROP, 'econ');

%% Find singular values
sig = diag(S);
 plot(sig/sum(sig), 'ro', 'Linewidth', [1.5])
 ylabel('Energy level');
 axis([0 80 0 0.15])
 set(gca, 'Fontsize', [13], 'Xtick', [0 20 40 60 80])
 xlabel('First 80 singular values');
 title('Energy spectrum');
 r = [4, 25, 70, 150, 300];
 reconstruct = U * S(:, 1:50) * V(:, 1:50)';

 %%
 figure(2)
```

```
faces = [10, 25, 70, 200];

for i = 1:length(faces)
   subplot(2,2, i);
   new_image_2 = uint8(reshape(reconstruct(:, faces(i)),243, 320)); %192, 168));
   imshow(new_image_2);
end

test_file = 'yalefaces_cropped/CroppedYale/yaleB01/yaleB01_P00A+000E+00.pgm';
test = imread(test_file);
figure(3)
subplot(2,3,1)
imshow(test);

title('original picture')
test = double(reshape(test, 192 *168, 1));
%%
%figure(4)
for i = 1:length(r)
   U_new = U(:, 1:r(i));
   new_image = U_new * U_new' * test;
   new_image_2 = reshape(new_image, 192, 168);
   subplot(2,3, i + 1);
   imshow(uint8(new_image_2));
   pic_title = ['rank = ', num2str(r(i))];
   title(pic_title)
end
%%
U_new = U(:, 1:4);
new_image = U_new * U_new' * test;
new_image_2 = reshape(new_image, 192, 168);
figure(5)
imshow(uint8(new_image_2));
pic_title = ['rank = 4'];
title(pic_title)

%%
figure(5)
for i = 1:4
    figure(i)
    ut1 = reshape(U(:,i), 192, 168);
    ut2=ut1(192:-1:1,:);
    pcolor(ut2)
    set(gca, 'Xtick', [], 'Ytick',[])
    title(['Principal component 0', num2str(i)])
end
%%
figure(2)
    ut1 = reshape(U(:,1), 192, 168);
    ut2=ut1(192:-1:1,:);
    pcolor(ut2)
    set(gca, 'Xtick', [], 'Ytick',[])
    title(['Principal component 0', num2str(1)])
```

```
     figure(3)
     plot(V(:,1))
     title('V vector: face 1')
     xlabel('feature points')
     ylabel('feature values')
     figure(4)
     plot(V(:,2))
     title('V vector: face 2')
     xlabel('feature points')
     ylabel('feature values')
%% uncropped

UNCROPPED = [];
ave_face = [];
mainFolder2 = dir('yalefaces_uncropped/yalefaces');
mainFolder2 = mainFolder2(3:length(mainFolder2));
for i=1:length(mainFolder2)
    name = mainFolder2(i).name;
    a = ['yalefaces_uncropped/yalefaces', '/', name];
    data = imread(a);
    data = reshape(data,243*320,1);
    UNCROPPED = [UNCROPPED data];
end
UNCROPPED = double(UNCROPPED);
[U,S,V] = svd(UNCROPPED, 'econ');

%% Find singular values
sig = diag(S);
 subplot(2,1,1)
 plot(sig/sum(sig), 'ro', 'Linewidth', [1.5])
 ylabel('Energy level');
 axis([0 80 0 0.2])
 set(gca, 'Fontsize', [13], 'Xtick', [0 20 40 60 80])
 xlabel('First 80 singular values');
 title('Energy spectrum of Uncropped pictures');
 r = [4, 25, 70, 150, 300];
 reconstruct = U * S(:, 1:50) * V(:, 1:50)';

 %%
 figure(3)
 test_file = 'yalefaces_uncropped/yalefaces/subject01.normal';
 test = imread(test_file);
 subplot(3,2, 1)
 imshow(test);
 title('original picture of subject01')
  test_file = 'yalefaces_uncropped/yalefaces/subject04.normal';
 test = imread(test_file);
 subplot(3,2, 3)
 imshow(test);
 title('original picture of subject04')
  test_file = 'yalefaces_uncropped/yalefaces/subject08.normal';
 test = imread(test_file);
 subplot(3,2, 5)
 imshow(test);
```

```
title('original picture of subject08')
%%
r = [6, 39, 83];
name = ['1', '4', '8'];
for i = 1:length(r)
   new_image_2 = uint8(reshape(reconstruct(:, r(i)),243, 320));
   subplot(3,2, 2*i)
   imshow(new_image_2);
   title_name = ['restored subject0', name(i)];
   title(title_name);
end

%% testing
figure(5)
for i = 1:4
    figure(i)
    ut1 = reshape(U(:,i), 243, 320);
    ut2=ut1(243:-1:1,:);
    pcolor(ut2)
    set(gca, 'Xtick', [], 'Ytick',[])
    title(['Principal component 0', num2str(i)])
end
clc; close all; clear all;

% path of each folder
billie_dir_string = ['Part2/Billie/'];
chopin_dir_string = ['Part2/Chopin/'];
mj_dir_string = ['Part2/MJ/'];
ari_dir_string = ['Part2/ari/'];
Classic_dir_string = ['Part2/Classic/'];
dpr_dir_string = ['Part2/dpr/'];
EDM_dir_string = ['Part2/EDM/'];


billie_dir = dir('Part2/Billie');
chopin_dir = dir('Part2/Chopin');
mj_dir = dir('Part2/MJ');
ari_dir = dir('Part2/ari');
Classic_dir = dir('Part2/Classic');
dpr_dir = dir('Part2/dpr');
EDM_dir = dir('Part2/EDM');
billie_dir = billie_dir(3:end);
chopin_dir = chopin_dir(3:end);
mj_dir = mj_dir(3:end);
ari_dir = ari_dir(3:end);
Classic_dir = Classic_dir(3:end);
dpr_dir = dpr_dir(3:end);
EDM_dir = EDM_dir(3:end);

%% pre-process billie songs
song_title = [billie_dir_string, billie_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(billie_dir)
    song_title = [billie_dir_string, billie_dir(i).name];
```

```
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save billie.dat result -ascii;
%[y,Fs] = audioread([billie_dir_string, billie_dir(1).name]);
%sound(y,Fs)

%% pre-process MJ songs
song_title = [mj_dir_string, mj_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(mj_dir)
    song_title = [mj_dir_string, mj_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save mj.dat result -ascii;

%% pre-process Chopin
song_title = [chopin_dir_string, chopin_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(chopin_dir)
    song_title = [chopin_dir_string, chopin_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save chopin.dat result -ascii;

%% pre-process ari songs
song_title = [ari_dir_string, ari_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(ari_dir)
    song_title = [ari_dir_string, ari_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save ari.dat result -ascii;

%%% pre-process Classic songs
song_title = [Classic_dir_string, Classic_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(Classic_dir)
    song_title = [Classic_dir_string, Classic_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save Classic.dat result -ascii;

%%% pre-process dpr songs
song_title = [dpr_dir_string, dpr_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(dpr_dir)
    song_title = [dpr_dir_string, dpr_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
```

```
end
save dpr.dat result -ascii;

%%% pre-process EDM songs
song_title = [EDM_dir_string, EDM_dir(1).name];
result = preprocessing(song_title);
for i = 2:length(EDM_dir)
    song_title = [EDM_dir_string, EDM_dir(i).name];
    result_temp = preprocessing(song_title);
    result = [result result_temp];
end
save EDM.dat result -ascii;

%% reshape to have same length for each category
clc; close all; clear all;
load ari.dat;load billie.dat;load chopin.dat;load Classic.dat;
load dpr.dat;load mj.dat;load EDM.dat;
ari = ari(:);
ari = ari(1:25137000);
ari = reshape(ari, 220500, 114);
ari = ari(:,1:90);
save ari.dat ari -ascii;
billie = billie(:);
billie = billie(1:19845000);
billie = reshape(billie, 220500, 90);
save billie.dat billie -ascii;
Classic = Classic(:, 1:90);
save Classic.dat Classic -ascii;
dpr = dpr(:);
dpr = dpr(1:19845000);
dpr = reshape(dpr, 220500, 90);
save dpr.dat dpr -ascii;
EDM = EDM(:, 1:90);
save EDM.dat EDM -ascii;
mj = mj(:, 1:90);
save mj.dat mj -ascii;

%% test 1 classification
clc; close all; clear all;
load chopin.dat;
load dpr.dat;load mj.dat;

% Classification set up
chopin = fft_with_song(chopin);
dpr = fft_with_song(dpr);
mj = fft_with_song(mj);
q1 = randperm(90);
q2 = randperm(90);
q3 = randperm(90);
x1 = chopin(:, q1(1:70));
x2 = dpr(:, q1(1:70));
x3 = mj(:, q1(1:70));

X = [x1 x2 x3]; %% 90 for each songs
```

```
[u s v] = svd(X, 'econ');
S = diag(s);
figure(1)

plot(S/sum(S), 'ro', 'LineWidth', [2])
xlabel('Song clips singular values')
ylabel('Singluar value energy level')
title('Singular value spectrum for test 1')
energy = sum(S(1:70))/sum(S);
%%
singular_values = 59;
projected_value = u(:,1:singular_values)' * [chopin dpr mj];
xtrain = [projected_value(:,q1(1:70)) projected_value(:, q2(1:70)+90) projected_value(:, q3(1:70)+180)]

c_train = [ones(70,1) ; 2 * ones(70,1) ; 3 * ones(70,1)];
true_value = [ones(20,1); 2* ones(20,1) ; 3 * ones(20,1)];
cross_validate = [];
past_success = 0;
times = 1000;
for j = 1: times
    q1 = randperm(90);
    q2 = randperm(90);
    q3 = randperm(90);
    past_success = 0;
    %xtrain = [projected_value(:,q1(1:70)) projected_value(:, q2(1:70)+90) projected_value(:, q3(1:70)+
    xtest = [projected_value(:,q1(71:90)) projected_value(:, q2(71:90)+90) projected_value(:, q3(71:90)+
    result=classify(xtest', xtrain', c_train);
    indeces = result == true_value;
    success =nnz(indeces) / 60 * 100;
    if past_success < success
        figure(2)
        bar(classify(xtest', xtrain', c_train))
        past_success = success;
    end
    cross_validate = [cross_validate, success];
end
average_success = sum(cross_validate) / 20;
figure(3)
hist(cross_validate)
xlabel('Success rate(%)');
ylabel('Number of cases');
title('Success rate distribution: test1')
pd = fitdist(cross_validate','Normal');
printing = ['The mean of ', num2str(times), 'cross-validation is ', num2str(pd.mu)];
printing2 = ['The std is ', num2str(pd.sigma)];
disp(printing);
disp(printing2);

%% test 2 classification
clc; close all; clear all;
load mj.dat;
load billie.dat;
load ari.dat
```

```
mj = fft_with_song(mj);
billie = fft_with_song(billie);
ari = fft_with_song(ari);
%%
q1 = randperm(90);
q2 = randperm(90);
q3 = randperm(90);
x1 = mj(:, q1(1:70));
x2 = billie(:, q1(1:70));
x3 = ari(:, q1(1:70));
X = [x1 x2 x3]; %% 90 for each songs
[u s v] = svd(X, 'econ');
S = diag(s);
figure(1)
plot(S/sum(S), 'ro', 'LineWidth', [2])
xlabel('Song clips singular values')
ylabel('Singluar value energy level')
title('Singular value spectrum for test 2')
energy = sum(S(1:70))/sum(S);
%%
singular_values = 76;
projected_value = u(:,1:singular_values)' * [mj billie ari];
xtrain = [projected_value(:,q1(1:70)) projected_value(:, q2(1:70)+90) projected_value(:, q3(1:70)+180)]

c_train = [ones(70,1) ; 2 * ones(70,1) ; 3 * ones(70,1)];
true_value = [ones(20,1); 2* ones(20,1) ; 3 * ones(20,1)];
cross_validate = [];
past_success = 0;
times = 1000;
for j = 1: times
    q1 = randperm(90);
    q2 = randperm(90);
    q3 = randperm(90);
    past_success = 0;
    %xtrain = [projected_value(:,q1(1:70)) projected_value(:, q2(1:70)+90) projected_value(:, q3(1:70)+
    xtest = [projected_value(:,q1(71:90)) projected_value(:, q2(71:90)+90) projected_value(:, q3(71:90)+
    result=classify(xtest', xtrain', c_train);
    indeces = result == true_value;
    success =nnz(indeces) / 60 * 100;
    if past_success < success
        figure(2)
        bar(classify(xtest', xtrain', c_train))
        past_success = success;
    end
    cross_validate = [cross_validate, success];
end
average_success = sum(cross_validate) / 20;

figure(3)
hist(cross_validate)
xlabel('Success rate(%)');
ylabel('Number of cases');
title('Success rate distribution: test2')
```

```
pd = fitdist(cross_validate','Normal');
printing = ['The mean of ', num2str(times), 'cross-validation is ', num2str(pd.mu)];
printing2 = ['The std is ', num2str(pd.sigma)];
disp(printing);
disp(printing2);

%% test 3 classification
clc; close all; clear all;
load ari.dat; load Classic.dat; load EDM.dat;
%%
% Classification set up
ari = fft_with_song(ari);
Classic = fft_with_song(Classic);
EDM = fft_with_song(EDM);
%%
q1 = randperm(90);
q2 = randperm(90);
q3 = randperm(90);
x1 = ari(:, q1(1:70));
x2 = Classic(:, q1(1:70));
x3 = EDM(:, q1(1:70));

X = [x1 x2 x3]; %% 90 for each songs

[u s v] = svd(X, 'econ');
S = diag(s);
figure(1)

plot(S/sum(S), 'ro', 'LineWidth', [2])
xlabel('Song clips singular values')
ylabel('Singluar value energy level')
title('Singular value spectrum for test 3')
energy = sum(S(1:70))/sum(S);
%%
singular_values = 70;
projected_value = u(:,1:singular_values)' * [ari Classic EDM];
xtrain = [projected_value(:,q1(1:70)) projected_value(:, q2(1:70)+90) projected_value(:, q3(1:70)+180)]

c_train = [ones(70,1) ; 2 * ones(70,1) ; 3 * ones(70,1)];
true_value = [ones(20,1); 2* ones(20,1) ; 3 * ones(20,1)];
cross_validate = [];
past_success = 0;
times = 1000;
for j = 1: times
    q1 = randperm(90);
    q2 = randperm(90);
    q3 = randperm(90);
    past_success = 0;
    xtest = [projected_value(:,q1(71:90)) projected_value(:, q2(71:90)+90) projected_value(:, q3(71:90)+
    result=classify(xtest', xtrain', c_train);
    indeces = result == true_value;
    success =nnz(indeces) / 60 * 100;
    if past_success < success
        figure(2)
```

```
        bar(classify(xtest', xtrain', c_train))
        past_success = success;
    end
    cross_validate = [cross_validate, success];
end
%%
figure(3)
hist(cross_validate)
xlabel('Success rate(%)');
ylabel('Number of cases');
title('Success rate distribution: test3')
pd = fitdist(cross_validate','Normal');
printing = ['The mean of ', num2str(times), 'cross-validation is ', num2str(pd.mu)];
printing2 = ['The std is ', num2str(pd.sigma)];
disp(printing);
disp(printing2);

function result = preprocessing(name_of_audio)
    [y, Fs] = audioread(name_of_audio);
    y = mean(y,2);
    [m,n]=size(y);
    dt = 1/Fs;
    t = dt*(0:m-1);
    temp = max(t)/5;
    rounded_number = floor(temp);
    t1 = 0;
    t2 = 5;
    idx = (t>=t1) & (t<t2);
    result = y(idx);
    for i = 2:rounded_number
        t1 = t1 + 5;
        t2 = t2 + 5;
        idx = (t>=t1) & (t<t2);
        result = [result y(idx)];
    end
end
function spec_result = fft_with_song(song)
    spec_result = [];
    for i = 1:90
        ft = fft(song(:,i));
        spec = abs(fftshift(ft));
        spec_result = [spec_result spec];
    end
    [m,n] = size(spec_result);
    mn = mean(spec_result,2);
    spec_result = spec_result-repmat(mn,1,n);
end
```