Homework Set Five
ECE 271A

Name : Hsiu-Wen Yen

ID : A59010599

To run EM algorithm, we need to choose random initial value as parameters. However, there are some conditions are required.

(1) Summation of probability of hidden classes is one.
(2) Covariance matrix must be square symmetric and positive definite matrix.

With all initial value, we can run EM algorithm to calculate parameters for P(X|Z).

For E-Step, we need to find hij matrix.

E-step:

$$h_{ij} = P_{\mathbf{Z}|\mathbf{X}}(\mathbf{e}_j|\mathbf{x}_i;\boldsymbol{\Psi}^{(n)}) = \frac{\mathcal{G}\left(\mathbf{x}_i,\mu_j^{(n)},\sigma_j^{(n)}\right)\pi_j^{(n)}}{\sum_{k=1}^{C}\mathcal{G}\left(\mathbf{x}_i,\mu_k^{(n)},\sigma_k^{(n)}\right)\pi_k^{(n)}}$$

Using formula above, we can get hij matrix during each iteration.

For M-Step, we need to update all parameter for next iteration.

- leads to the update equations

$$\mu_j^{(n+1)} = \frac{\sum_i h_{ij}\mathbf{x}_i}{\sum_i h_{ij}} \qquad \pi_j^{(n+1)} = \frac{1}{n}\sum_i h_{ij}$$

$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij}(\mathbf{x}_i - \mu_j)^2}{\sum_i h_{ij}}$$

Using formula above, we can calculate parameters.

However, we need to make sure covariance matrix is square symmetric and positive definite matrix, I add 0.0001 in each diagonal value to prevent 0.

To stop iteration, I check data likelihood difference during each iteration. if the absolute value of data likelihood difference between (n) and (n-1) is less than 0.0001, then I stop the iteration.

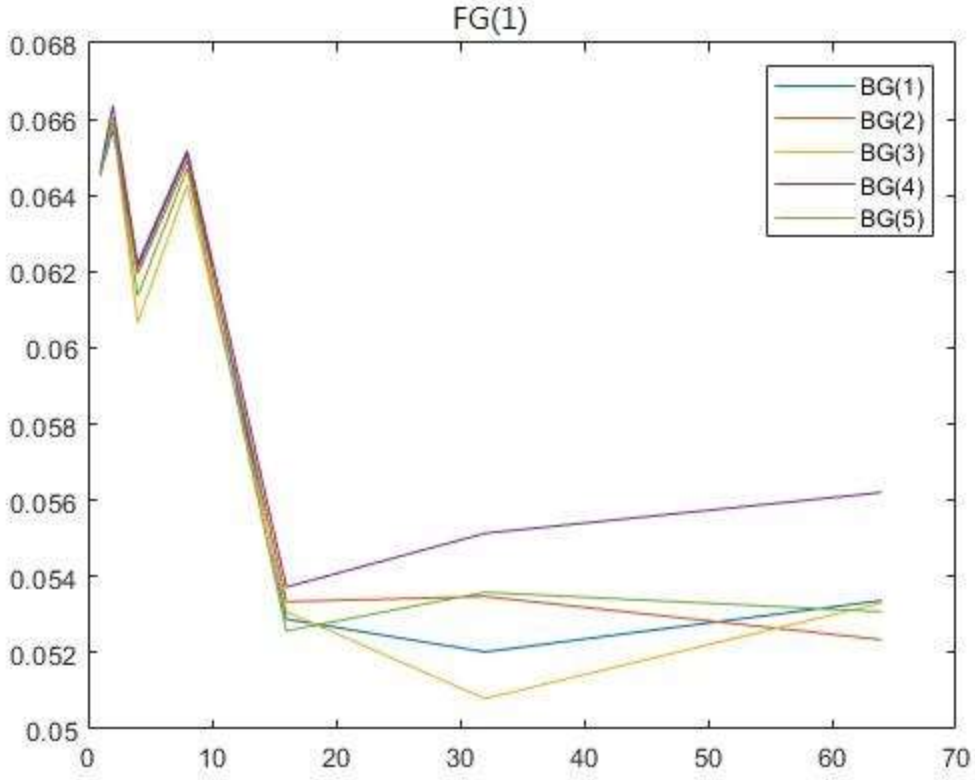
After all training procedure, we can plug all this parameter into BDR to do classification.

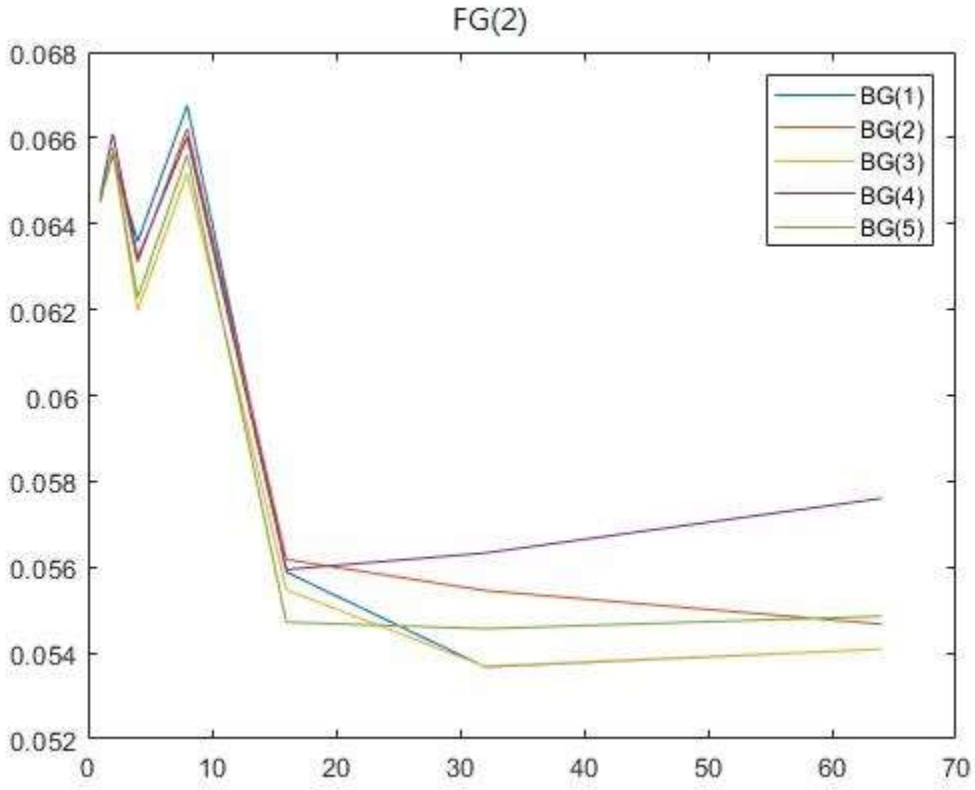
(a)

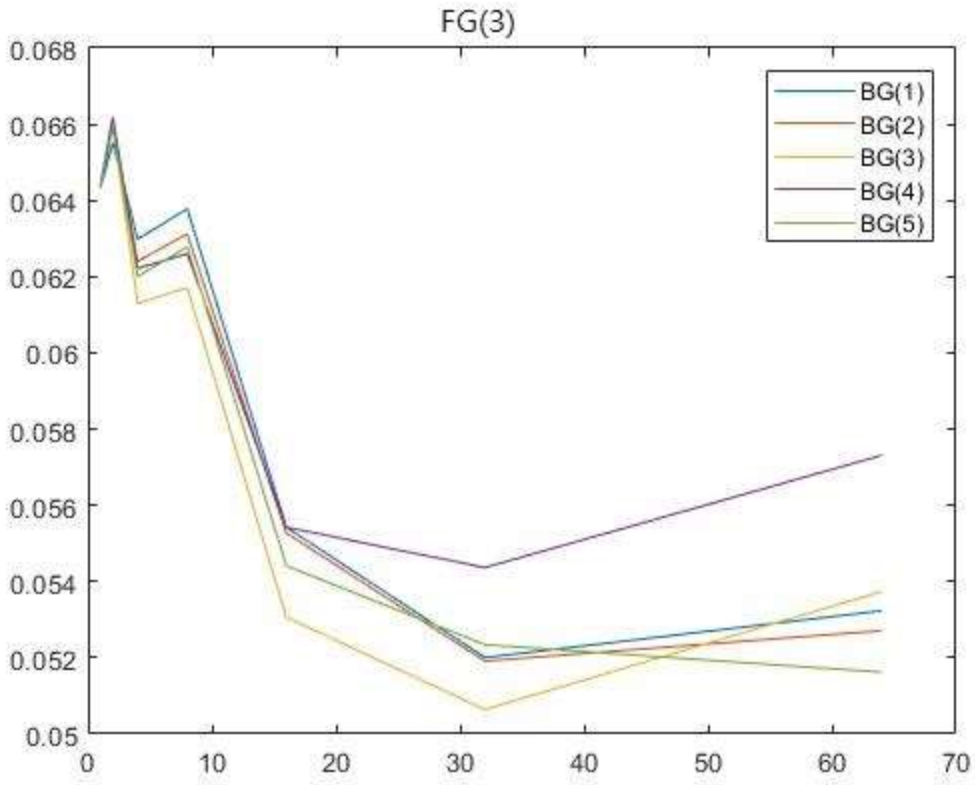
Train 5 mixture model for FG and 5 mixture model for BG and use different dimension to observe how EM algorithm work.

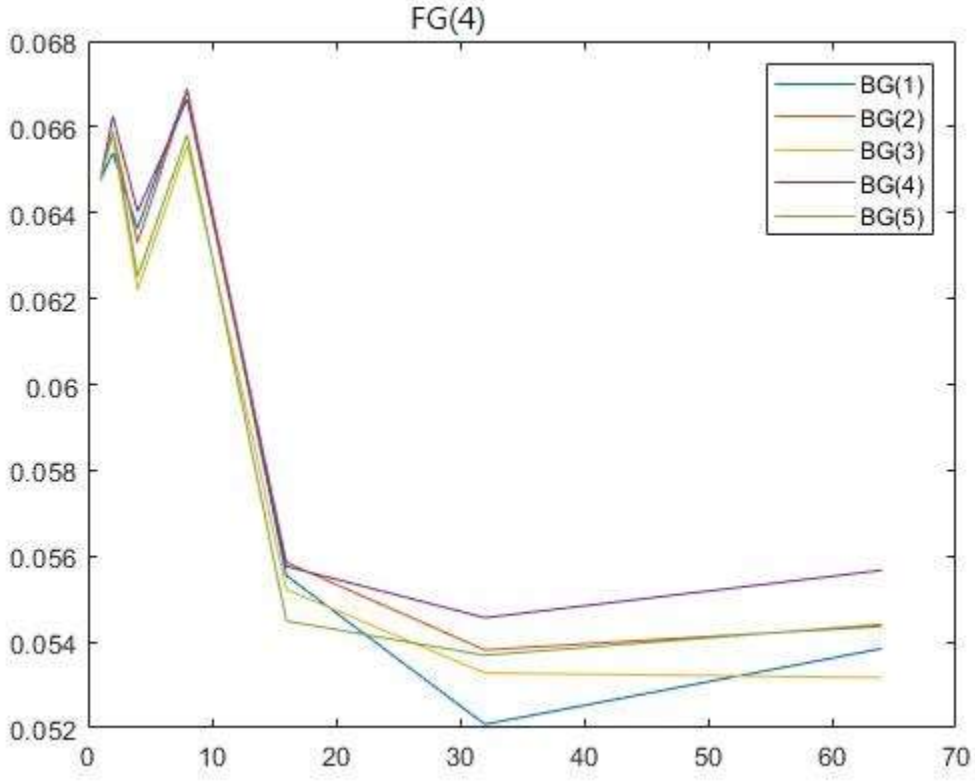
X axis is number of dimensions.

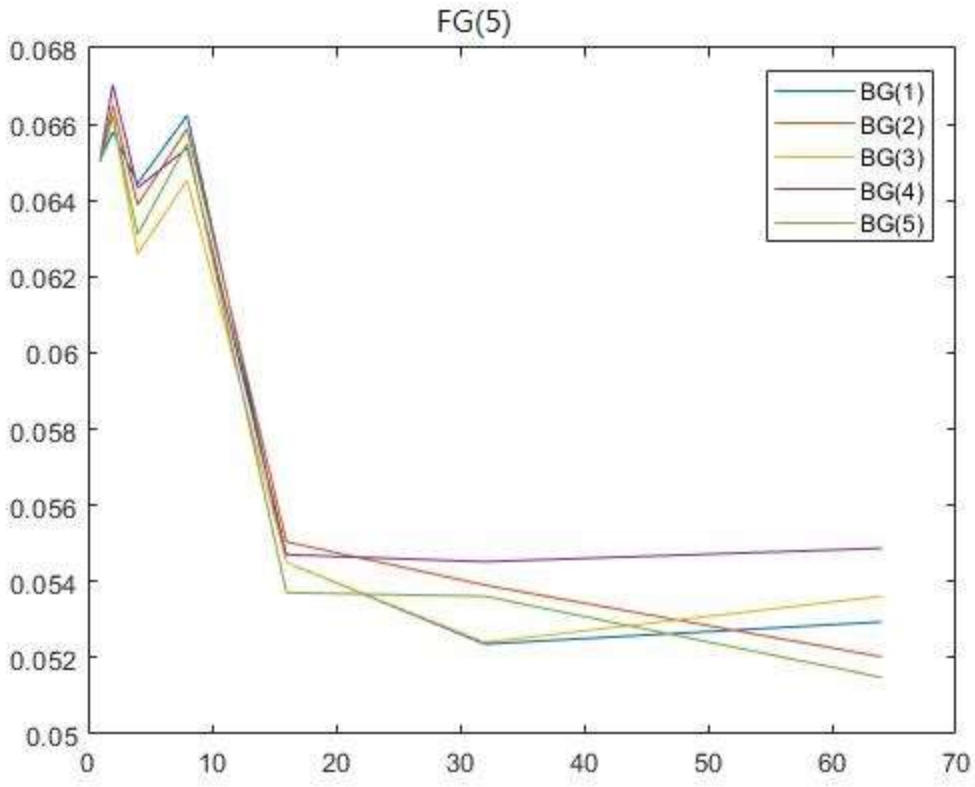
Y axis is Probability of error.


FG(1)
0.068
0.066
0.064
0.062
0.06
0.058
0.056
0.054
0.052
0.05
0
10
20
30
40
50
60
70
BG(1)
BG(2)
BG(3)
BG(4)
BG(5)


FG(2)
0.068
0.066
0.064
0.062
0.06
0.058
0.056
0.054
0.052
0
10
20
30
40
50
60
70
BG(1)
BG(2)
BG(3)
BG(4)
BG(5)


FG(3)
0.068
0.066
0.064
0.062
0.06
0.058
0.056
0.054
0.052
0.05
0
10
20
30
40
50
60
70
BG(1)
BG(2)
BG(3)
BG(4)
BG(5)

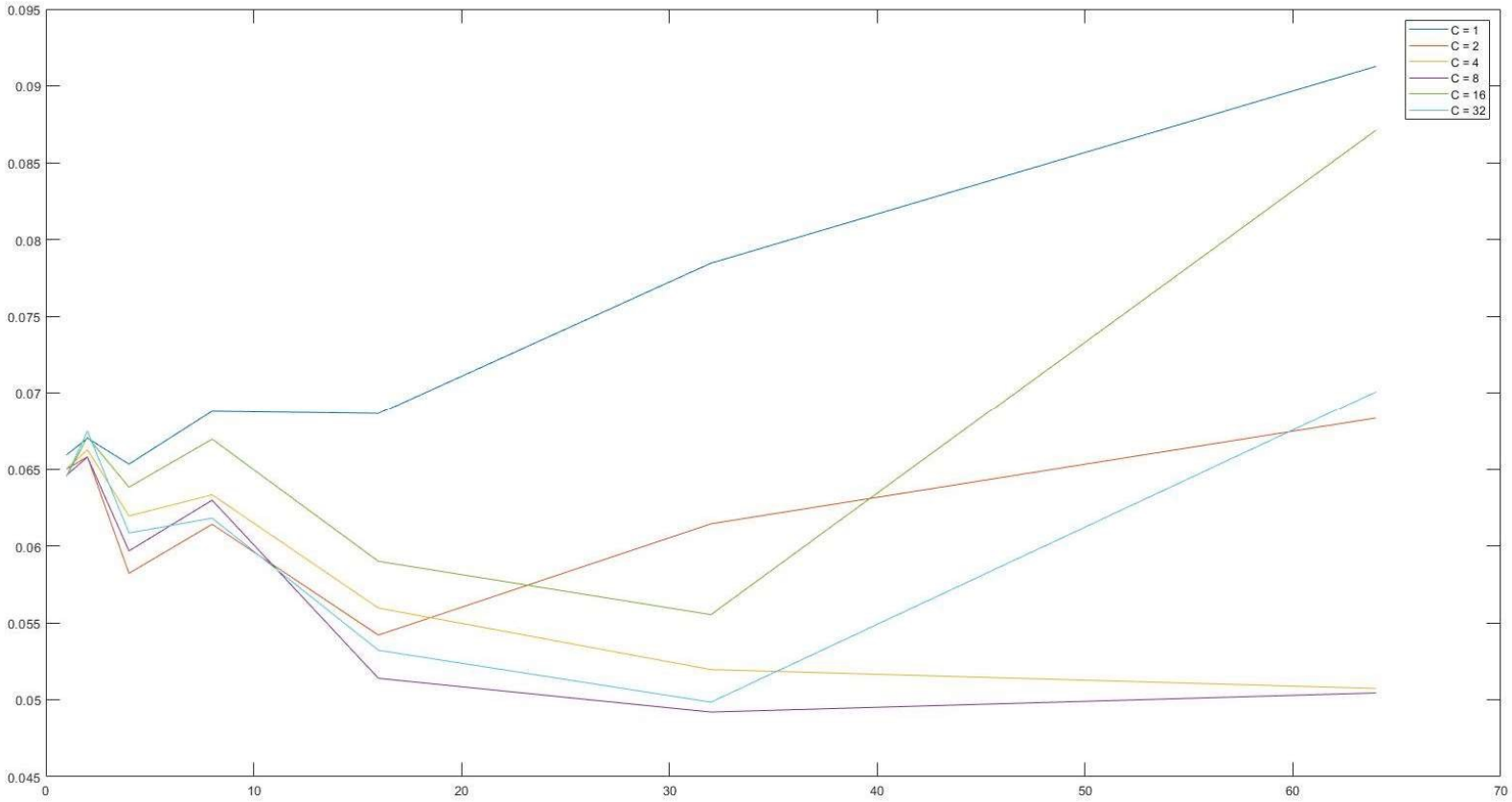

FG(4)
0.068
0.066
0.064
0.062
0.06
0.058
0.056
0.054
0.052
0
10
20
30
40
50
60
70
BG(1)
BG(2)
BG(3)
BG(4)
BG(5)



It is obvious that 25 curves are quite similar, it means that even with different initial value, EM algorithm will generate similar results. All the probability of error start near 0.065, and then quickly reduce to 0.055 when we use 16 features. Using more feature, we can see probability of error start to be different. Sometimes, it will get worse result than 16 feature. I think the reason for this behavior is overfitting.

Therefore, it means that EM algorithm performs well when we only use 16 features.

(b)



It's obvious that when dimension is 1, the results of different C are almost same. However, when dimension increase, different C start to act differently. It is interesting that when C increase, the dimension required to perform well decrease. For example, when dimension is set to 16, the slopes of probability of error are negative except C = 32. The best performance is C = 8 when dimension is 32.

This means that a good way to enhance performance is to increase the number of components, however, too many components may be harmful for your result. I think this is also kind of overfitting problem. Therefore, finding a specific number of components and dimension is key to make classification problem better. For different problem, we should find different coefficients because hidden variable will become different too.

Code Review

```
clear;clc;
% load trainging sample and image
load('TrainingSamplesDCT_8_new.mat');
cheetah = imread('cheetah.bmp');
cheetah=double(cheetah)/255;

% create 5 initial value for BG and FG and then run EM
%% 1th
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_BG);
[BG_pi_1, BG_mu_1, BG_cov_1] = runEM(TrainsampleDCT_BG, p_pi, mu, covariance);
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_FG);
[FG_pi_1, FG_mu_1, FG_cov_1] = runEM(TrainsampleDCT_FG, p_pi, mu, covariance);
%% 2th
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_BG);
[BG_pi_2, BG_mu_2, BG_cov_2] = runEM(TrainsampleDCT_BG, p_pi, mu, covariance);
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_FG);
[FG_pi_2, FG_mu_2, FG_cov_2] = runEM(TrainsampleDCT_FG, p_pi, mu, covariance);
%% 3th
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_BG);
[BG_pi_3, BG_mu_3, BG_cov_3] = runEM(TrainsampleDCT_BG, p_pi, mu, covariance);
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_FG);
[FG_pi_3, FG_mu_3, FG_cov_3] = runEM(TrainsampleDCT_FG, p_pi, mu, covariance);
%% 4th
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_BG);
[BG_pi_4, BG_mu_4, BG_cov_4] = runEM(TrainsampleDCT_BG, p_pi, mu, covariance);
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_FG);
[FG_pi_4, FG_mu_4, FG_cov_4] = runEM(TrainsampleDCT_FG, p_pi, mu, covariance);
%% 5th
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_BG);
[BG_pi_5, BG_mu_5, BG_cov_5] = runEM(TrainsampleDCT_BG, p_pi, mu, covariance);
[p_pi, mu, covariance] = Initialization_Value(TrainsampleDCT_FG);
[FG_pi_5, FG_mu_5, FG_cov_5] = runEM(TrainsampleDCT_FG, p_pi, mu, covariance);
%% combine them
FG_pi_set = {FG_pi_1,FG_pi_2,FG_pi_3,FG_pi_4,FG_pi_5};
BG_pi_set = {BG_pi_1,BG_pi_2,BG_pi_3,BG_pi_4,BG_pi_5};
FG_mu_set = {FG_mu_1,FG_mu_2,FG_mu_3,FG_mu_4,FG_mu_5};
BG_mu_set = {BG_mu_1,BG_mu_2,BG_mu_3,BG_mu_4,BG_mu_5};
FG_cov_set = {FG_cov_1,FG_cov_2,FG_cov_3,FG_cov_4,FG_cov_5};
BG_cov_set = {BG_cov_1,BG_cov_2,BG_cov_3,BG_cov_4,BG_cov_5};
%%
dimension_set = [1 2 4 8 16 32 64];

for fixed_FG = 1:5
        FG_pi = cell2mat(FG_pi_set(fixed_FG));
        FG_mu = cell2mat(FG_mu_set(fixed_FG));
        FG_cov = cell2mat(FG_cov_set(fixed_FG));
        figure(fixed_FG);
        txt = "FG mixture_ " + int2str(fixed_FG) +" comparision ";
        title(txt);
    for fix_BG = 1:5
        BG_pi = cell2mat(BG_pi_set(fix_BG));
        BG_mu = cell2mat(BG_mu_set(fix_BG));
```

```
        BG_cov = cell2mat(BG_cov_set(fix_BG));
storage = zeros(1,7);
for di_counter = 1 : 7
d = dimension_set(di_counter);
% calculate prior probabilities of cheetah and grass
pixel_total_count = size(TrainsampleDCT_FG, 1) + size(TrainsampleDCT_BG, 1);
prior_Pcheetah = size(TrainsampleDCT_FG, 1) / pixel_total_count;
prior_Pgrass = size(TrainsampleDCT_BG, 1) / pixel_total_count;

% create output mask image array
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A_64 = zeros(row_size, column_size);

% using 8 * 8 blocks to represent the left top pixel
for rows = 1 : row_size - 8 + 1
    for columns = 1 : column_size - 8 + 1
        block = cheetah(rows:rows+7, columns:columns+7);
        block = dct2(block);
        x = expand_zigzag(block);
        % for FG and BG
        p_FG = 0;
        for counter = 1 : 8
            p_FG = p_FG + mvnpdf(x(1:d), FG_mu(counter,1:d), FG_cov(1:d,1:d,counter))
* FG_pi(counter);
        end
        p_FG = p_FG * prior_Pcheetah;

        p_BG = 0;
        for counter = 1 : 8
            p_BG = p_BG + mvnpdf(x(1:d), BG_mu(counter,1:d), BG_cov(1:d,1:d,counter))
* BG_pi(counter);
        end
        p_BG = p_BG * prior_Pgrass;

        if (p_BG > p_FG)
            A_64(rows, columns) = 0;
        else
            A_64(rows, columns) = 1;
        end
    end
end
%figure();
%imagesc(A_64);
%colormap(gray(255));
%% error
% load cheetah mask.bmp
truth = imread("cheetah_mask.bmp");
% calculate last meaningful index of row and column
last_row = size(cheetah, 1) - 8 + 1;
last_column = size(cheetah, 2) - 8 + 1;
% error for 64d
truth = double(truth(1 : last_row, 1 : last_column) / 255);
A_64 = A_64(1 : last_row, 1 : last_column);
err = truth - A_64;
```

```
err = abs(err);
probability_error_64d = sum(err,'all') / (last_row*last_column);
storage(di_counter) = probability_error_64d;
end
plot(dimension_set, storage);
hold on;
    end
    hold off;
    legend('BG(1)','BG(2)','BG(3)','BG(4)','BG(5)');
end

%% run for different C
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,1);
[BG_pi_1, BG_mu_1, BG_cov_1] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
1);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,2);
[BG_pi_2, BG_mu_2, BG_cov_2] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
2);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,4);
[BG_pi_4, BG_mu_4, BG_cov_4] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
4);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,8);
[BG_pi_8, BG_mu_8, BG_cov_8] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
8);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,16);
[BG_pi_16, BG_mu_16, BG_cov_16] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
16);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_BG,32);
[BG_pi_32, BG_mu_32, BG_cov_32] = runEMwithC(TrainsampleDCT_BG, p_pi, mu, covariance,
32);

%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 1);
[FG_pi_1, FG_mu_1, FG_cov_1] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
1);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 2);
[FG_pi_2, FG_mu_2, FG_cov_2] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
2);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 4);
[FG_pi_4, FG_mu_4, FG_cov_4] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
4);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 8);
[FG_pi_8, FG_mu_8, FG_cov_8] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
8);
%%
clc;
```

```
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 16);
[FG_pi_16, FG_mu_16, FG_cov_16] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
16);
%%
[p_pi, mu, covariance] = Initialization_ValueWithC(TrainsampleDCT_FG, 32);
[FG_pi_32, FG_mu_32, FG_cov_32] = runEMwithC(TrainsampleDCT_FG, p_pi, mu, covariance,
32);

%% combine them
FG_pi_set = {FG_pi_1,FG_pi_2,FG_pi_4,FG_pi_8,FG_pi_16,FG_pi_32};
BG_pi_set = {BG_pi_1,BG_pi_2,BG_pi_4,BG_pi_8,BG_pi_16,BG_pi_32};
FG_mu_set = {FG_mu_1,FG_mu_2,FG_mu_4,FG_mu_8,FG_mu_16,FG_mu_32};
BG_mu_set = {BG_mu_1,BG_mu_2,BG_mu_4,BG_mu_8,BG_mu_16,BG_mu_32};
FG_cov_set = {FG_cov_1,FG_cov_2,FG_cov_4,FG_cov_8,FG_cov_16,FG_cov_32};
BG_cov_set = {BG_cov_1,BG_cov_2,BG_cov_4,BG_cov_8,BG_cov_16,BG_cov_32};
%%
C_list = [1 2 4 8 16 32];
figure();

for c_choosen = 1 : 6
c = C_list(c_choosen);

FG_pi = cell2mat(FG_pi_set(c_choosen));
FG_mu = cell2mat(FG_mu_set(c_choosen));
FG_cov = cell2mat(FG_cov_set(c_choosen));
BG_pi = cell2mat(BG_pi_set(c_choosen));
BG_mu = cell2mat(BG_mu_set(c_choosen));
BG_cov = cell2mat(BG_cov_set(c_choosen));
storage = zeros(1,7);
for di_counter = 1 : 7
d = dimension_set(di_counter);
% calculate prior probabilities of cheetah and grass
pixel_total_count = size(TrainsampleDCT_FG, 1) + size(TrainsampleDCT_BG, 1);
prior_Pcheetah = size(TrainsampleDCT_FG, 1) / pixel_total_count;
prior_Pgrass = size(TrainsampleDCT_BG, 1) / pixel_total_count;

% create output mask image array
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A_64 = zeros(row_size, column_size);

% using 8 * 8 blocks to represent the left top pixel
for rows = 1 : row_size - 8 + 1
    for columns = 1 : column_size - 8 + 1
        block = cheetah(rows:rows+7, columns:columns+7);
        block = dct2(block);
        x = expand_zigzag(block);
        % for FG and BG
        p_FG = 0;
        for counter = 1 : c
            p_FG = p_FG + mvnpdf(x(1:d), FG_mu(counter,1:d), FG_cov(1:d,1:d,counter))
* FG_pi(counter);
        end
        p_FG = p_FG * prior_Pcheetah;
```

```
        p_BG = 0;
        for counter = 1 : c
            p_BG = p_BG + mvnpdf(x(1:d), BG_mu(counter,1:d), BG_cov(1:d,1:d,counter))
* BG_pi(counter);
        end
        p_BG = p_BG * prior_Pgrass;

        if (p_BG > p_FG)
            A_64(rows, columns) = 0;
        else
            A_64(rows, columns) = 1;
        end
    end
end
%figure();
%imagesc(A_64);
%colormap(gray(255));
%% error
% load cheetah mask.bmp
truth = imread("cheetah_mask.bmp");
% calculate last meaningful index of row and column
last_row = size(cheetah, 1) - 8 + 1;
last_column = size(cheetah, 2) - 8 + 1;
% error for 64d
truth = double(truth(1 : last_row, 1 : last_column) / 255);
A_64 = A_64(1 : last_row, 1 : last_column);
err = truth - A_64;
err = abs(err);
probability_error_64d = sum(err,'all') / (last_row*last_column);
storage(di_counter) = probability_error_64d;
end
plot(dimension_set, storage);
hold on;
end
hold off;
legend('C = 1','C = 2','C = 4','C = 8','C = 16','C = 32');
```

```
function [p_pi, mu, covariance] = Initialization_ValueWithC(data,c)
    p_pi = rand(1, c);
    p_pi = p_pi / sum(p_pi); % initial pi

    % pick one data from dataset as initial value for mu

    mu = zeros(c, 64);
    for counter = 1 : c
      mu(counter,:) = data(randi(size(data,1)),:);
    end

    covariance = zeros(64, 64, c);
    for counter = 1 : c
        covariance(:,:,counter) = (rand(1, 64)).* eye(64);
    end

end
```

```
function [new_pi, new_mu, new_cov, time] = runEMwithC(data, pre_pi, pre_mu, pre_cov,
c)

for time = 1 : 1000
% create h
h = zeros(size(data,1), c);
BDRjoint=zeros(size(data,1), c);
% E-step
% calculate h
for i = 1 : size(data,1)
    for j = 1:c
        BDRjoint(i,j) = mvnpdf(data(i,:),pre_mu(j,:),pre_cov(:,:,j)) * pre_pi(j);
    end
    h_denominator = sum(BDRjoint(i,:));
    h(i,:) = BDRjoint(i,:) / h_denominator;
end
BDRlikely(time) = sum( log( sum(BDRjoint,2) )  );

% M-step
% update paremeters
new_pi = sum(h) / size(data,1);

%update for mu
new_mu = zeros(c, 64);
for j = 1 : c

    tem_data = data;
    for i = 1 : size(data, 1)
        tem_data(i,:) = tem_data(i,:) * h(i,j);
    end
    new_mu(j,:) = sum(tem_data) / sum(h(:,j));
end

% update for covariance
new_cov = zeros(64,64,c);

for j = 1 : c
    new_cov(:,:,j) = diag(diag(( (data-pre_mu(j,:))' .*h(:,j)'*(data-
pre_mu(j,:))./sum(h(:,j),1))+0.0001));
end

pre_pi = new_pi;
pre_mu = new_mu;
pre_cov = new_cov;

if (time > 1)
    if (abs(BDRlikely(time) - BDRlikely(time-1))<0.001)
        break;
    end
end

end

end
```

```
function myArray = expand_zigzag(matrix)
```

```
    load("Zig-Zag Pattern.txt");
    myArray = zeros(1, 64);
    for row = 1 : size(matrix,1)
        for column = 1 : size(matrix,2)
            number = Zig_Zag_Pattern(row, column) + 1;
            myArray(number) = matrix(row, column);
        end
    end
end
```