Quiz 5 Report

a. In part a, the goal is to learn 5 gaussian mixtures models of 8 components with EM and evaluate the performance of the 25 classifiers. Firstly, π_c, Σ_c, μ_c are randomly initialized. It should be noticed that π_c should be normalized and Σ_c is the diagonal covariance.

In E step, the $h_{ij} = P_{Z|X}(x_i|e_j;\psi^{(n)}) = \frac{g(x_i,\mu_j^{(n)},\sigma_j^{(n)})\pi_j^{(n)}}{\sum_{k=1}^{C} g(x_i,\mu_k^{(n)},\sigma_k^{(n)})\pi_k^{(n)}}$.

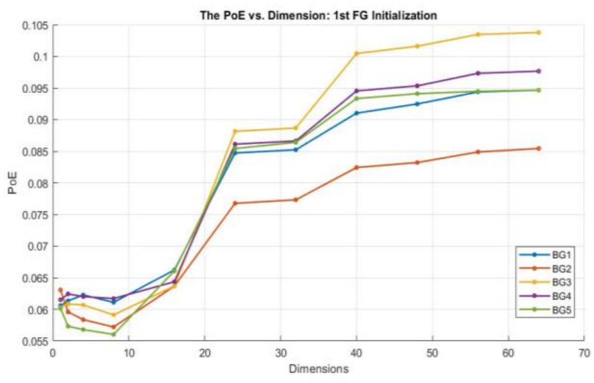
In M step, the π_c, Σ_c, μ_c are updated by:

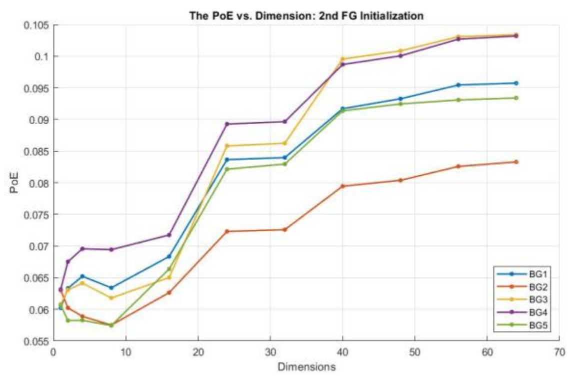
$$\pi_j^{(n+1)} = \frac{1}{n}\sum_i h_{ij};$$

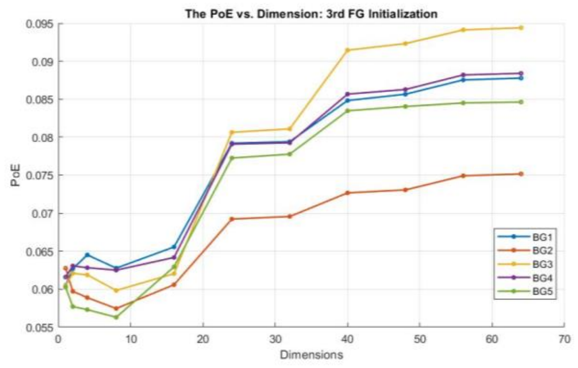
$$\mu_j^{(n+1)} = \frac{\sum_i h_{ij}x_i}{\sum_i h_{ij}};$$

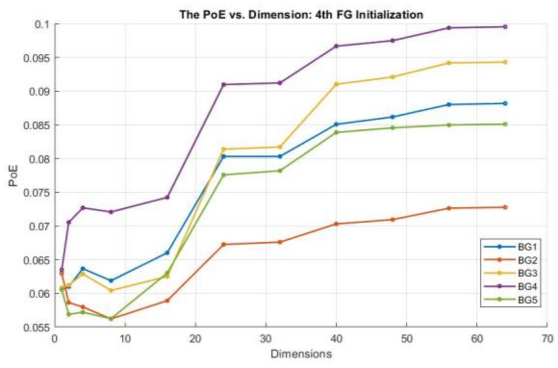
$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij}(x_i-\mu_j)^2}{\sum_i h_{ij}}$$

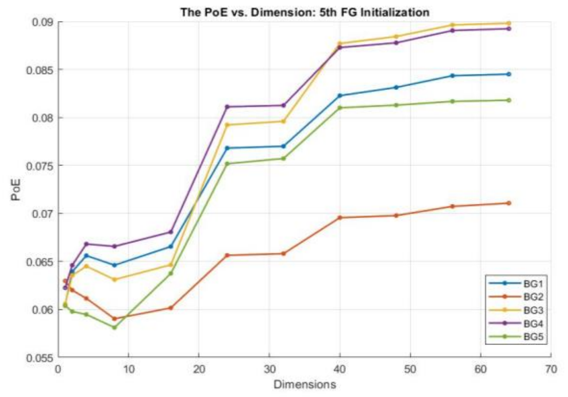
Based on that, the plots of the probability of error vs. dimensions of all 5 gaussian mixtures with 25 classifiers are shown below. From the plots, the probability of error of all classifiers starts at around 0.06 when dimension equals to 1 and decreases to around 0.057 when dimension equals to 8. As the dimension increases after it reached to 8, the probability of error of all classifiers rapidly increases. Recall to the homework 2, we know that introducing high dimensions to model may not bring a positive impact, which can be observed from the tendency of the probability of error of the classifiers as the dimension increases in this part. Besides, the classifier with the best overall performance is the classifier of the 5th FG and the 2nd BG initializations. The probability of error of this classifier increases smoothly as the dimension gets higher. The classifier of the 1st FG and 5th BG initializations has the lowest probability of error at dimension equals to 8. Overall, the performances of all 25 classifiers do not have a distinct gap when dimension is low (from 1 to 8). It should be noticed that the dimensions introduced into the model are not selected by any wiser method rather than the order of them. If the used dimensions are selected wisely, we should expect a better performance and the more distinct difference of performance of the mixtures.



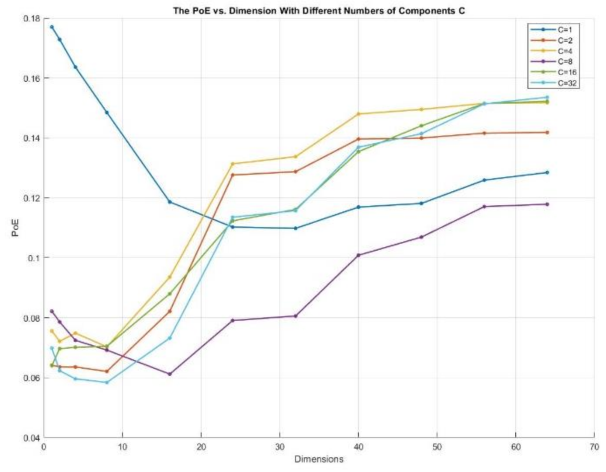








b. In this part, instead of evaluating the performance of the classifiers of different mixtures within the same number of components, the performance of classifiers with different number of components (C = {1,2,4,8,16,32}), which is shown below. Increasing the number of mixture components indeed improve the overall performance of the classifiers, especially when the number of components is low (i.e., from 1 to 8). When the number of components is relatively high (such as 16, 32), increases number of components helps little in improvement of the performance, while even brings negative impact on the performance, such as the case of 16 components shown below.



Code

Part a)

```
clear;
clc;
load('TrainingSamplesDCT_8_new.mat');
FG = TrainsampleDCT_FG;
BG = TrainsampleDCT_BG;

sample_BG = size(BG,1);
sample_FG = size(FG,1);

feature = size(BG,2);

img = im2double(imread('cheetah.bmp'));
mask = im2double(imread('cheetah_mask.bmp'));

%read Zig-Zag Pattern.txt file
zz = fopen('Zig-Zag Pattern.txt','r');
zzPat = fscanf(zz,'%d',[8,8])+1;
fclose(zz);

% obtain the DCT of the image
[row,colm] = size(img);
img_zzs = zeros(row-8,colm-8,64);
for i = 1:row-8
    for j = 1:colm-8
        dctImg = dct2(img(i:i+7,j:j+7));
        for x = 1:8
            for y = 1:8
                img_zzs(i,j,zzPat(x,y)) = dctImg(x,y);
            end
        end
    end
end
[r,m] = size(img_zzs,1,2);

%% Part a)
C = 8;
dimensions = [1,2,4,8,16,24,32,40,48,56,64];
PY_BG = sample_BG/(sample_FG+sample_BG);
PY_FG = sample_FG/(sample_FG+sample_BG);

% ****************************BG EM*****************************************
pi_BG = zeros(C,5);
mu_BG = zeros(C,feature,5);
sigma_BG = zeros(feature,feature,C,5);

for h = 1:5
    pi = randi(1,C);
    pi = pi./sum(pi);
    mu = BG(randperm(sample_BG,C),:);
    sigma = zeros(feature,feature,C);
    for i = 1:C
        sigma(:,:,i) = (rand(1,feature)).*eye(feature);
    end
    [pi,mu,sigma] = EM(C,BG,pi,mu,sigma);

    pi_BG(:,h) = pi;
    mu_BG(:,:,h) = mu;
    sigma_BG(:,:,:,h) = sigma;
end

% ***************************FG EM****************************************
% Initialization
pi_FG = zeros(C,5);
mu_FG = zeros(C,feature,5);
sigma_FG = zeros(feature,feature,C,5);

for h = 1:5
    pi = randi(1,C);
    pi = pi./sum(pi);
    mu = FG(randperm(sample_FG,C),:);
    sigma = zeros(feature,feature,C);

    for i = 1:C
        sigma(:,:,i) = (rand(1,feature)).*eye(feature);
    end
    [pi,mu,sigma] = EM(C,FG,pi,mu,sigma);

    pi_FG(:,h) = pi;
    mu_FG(:,:,h) = mu;
    sigma_FG(:,:,:,h) = sigma;
end
%%
% *************Calculate the Probability of FG and BG for BDR*************
PX_BG = zeros(r,m,length(dimensions),5);
PX_FG = zeros(r,m,length(dimensions),5);

PX_BG(:,:,:,1) = calPX(C,img_zzs,pi_BG(:,1),mu_BG(:,:,1),sigma_BG(:,:,:,1),PY_BG);
PX_FG(:,:,:,1) = calPX(C,img_zzs,pi_FG(:,1),mu_FG(:,:,1),sigma_FG(:,:,:,1),PY_FG);

PX_BG(:,:,:,2) = calPX(C,img_zzs,pi_BG(:,2),mu_BG(:,:,2),sigma_BG(:,:,:,2),PY_BG);
PX_FG(:,:,:,2) = calPX(C,img_zzs,pi_FG(:,2),mu_FG(:,:,2),sigma_FG(:,:,:,2),PY_FG);

PX_BG(:,:,:,3) = calPX(C,img_zzs,pi_BG(:,3),mu_BG(:,:,3),sigma_BG(:,:,:,3),PY_BG);
PX_FG(:,:,:,3) = calPX(C,img_zzs,pi_FG(:,3),mu_FG(:,:,3),sigma_FG(:,:,:,3),PY_FG);

PX_BG(:,:,:,4) = calPX(C,img_zzs,pi_BG(:,4),mu_BG(:,:,4),sigma_BG(:,:,:,4),PY_BG);
PX_FG(:,:,:,4) = calPX(C,img_zzs,pi_FG(:,4),mu_FG(:,:,4),sigma_FG(:,:,:,4),PY_FG);

PX_BG(:,:,:,5) = calPX(C,img_zzs,pi_BG(:,5),mu_BG(:,:,5),sigma_BG(:,:,:,5),PY_BG);
PX_FG(:,:,:,5) = calPX(C,img_zzs,pi_FG(:,5),mu_FG(:,:,5),sigma_FG(:,:,:,5),PY_FG);
%%
%***********************Calculate the PoE******************************
error_EM = zeros(25,length(dimensions));
% Set 1
error_EM(1,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,1));
```

```matlab
error_EM(2,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,2));
error_EM(3,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,3));
error_EM(4,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,4));
error_EM(5,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,5));

% Set 2
error_EM(6,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,1));
error_EM(7,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,2));
error_EM(8,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,3));
error_EM(9,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,4));
error_EM(10,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,5));

% Set 3
error_EM(11,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,1));
error_EM(12,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,2));
error_EM(13,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,3));
error_EM(14,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,4));
error_EM(15,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,5));

% Set 4
error_EM(16,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,1));
error_EM(17,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,2));
error_EM(18,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,3));
error_EM(19,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,4));
error_EM(20,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,5));

% Set 5
error_EM(21,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,1));
error_EM(22,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,2));
error_EM(23,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,3));
error_EM(24,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,4));
error_EM(25,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,5));

%%
% **************Generate the plot of the PoE vs. Dimensions***************
figure
hold on;
plot(dimensions,error_EM(1,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(2,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(3,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(4,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(5,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('BG1','BG2','BG3','BG4','BG5');
title('The PoE vs. Dimension: 1st FG Initialization');
xlabel('Dimensions');
ylabel('PoE')
grid on;
hold off;

figure(2)
hold on;
plot(dimensions,error_EM(6,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(7,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(8,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(9,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(10,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('BG1','BG2','BG3','BG4','BG5');
title('The PoE vs. Dimension: 2nd FG Initialization');
xlabel('Dimensions');
ylabel('PoE');
grid on;
hold off;

figure(3)
hold on;
plot(dimensions,error_EM(11,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(12,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(13,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(14,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(15,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('BG1','BG2','BG3','BG4','BG5');
title('The PoE vs. Dimension: 3rd FG Initialization');
xlabel('Dimensions');
ylabel('PoE');
grid on;
hold off;

figure(4)
hold on;
plot(dimensions,error_EM(16,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(17,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(18,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(19,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(20,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('BG1','BG2','BG3','BG4','BG5');
title('The PoE vs. Dimension: 4th FG Initialization');
xlabel('Dimensions');
ylabel('PoE');
grid on;
hold off;

figure(5)
hold on;
plot(dimensions,error_EM(21,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(22,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(23,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(24,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(25,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('BG1','BG2','BG3','BG4','BG5');
title('The PoE vs. Dimension: 5th FG Initialization');
xlabel('Dimensions');
ylabel('PoE');
grid on;
hold off;
%%
function [Pi_n,mu_n,sigma_n] = EM(C,X,Pi,mu,sigma)
    iter = 100; % maximum iteration
    likehood = zeros(1,iter); % log likehood for stopping EM
    jointpdf = zeros(size(X,1),C);
    mu_n = zeros(C,size(X,2));
    sigma_n = zeros(size(X,2),size(X,2),C);
    % EM
    for h = 1:iter
        % check the log likehood to stop the function once meet the
        % condition
        if h > 1
            if(abs((likehood(h)-likehood(h-1))/likehood(h))<0.01)
                break;
            end
        end

        % E-step
        for i = 1:C
            for j = 1:size(X,1)
                jointpdf(j,i) = sqrt((2*pi)^64*det(sigma(:,:,i)))^(-1)*exp(-(X(j,:)-
mu(i,:))/sigma(:,:,i)*(X(j,:)-mu(i,:))'/2)*Pi(i);
            end
        end
        hij = jointpdf ./ sum(jointpdf,2);

        % M-step
        Pi_n = sum(hij,1)/size(X,1);
        for i = 1:C
            mu_temp = 0;
            sigma_temp = 0;
            for j = 1:size(X,1)
                mu_temp = mu_temp + hij(j,i)*X(j,:);
                sig_temp = diag((X(j,:)-mu(i,:))'*(X(j,:)-mu(i,:)));
                sig_temp(sig_temp<1e-5) = 1e-5;
                sigma_temp = sigma_temp + hij(j,i)*diag(sig_temp);
            end
            mu_n(i,:) = mu_temp/sum(hij(:,i),1);
            sigma_n(:,:,i) = sigma_temp/sum(hij(:,i),1);
        end
    end
end

function px = calPX(C,X,Pi,mu,sigma,PY)
    px = zeros(247,262,11);
    dimensions = [1,2,4,8,16,24,32,40,48,56,64];
    for i = 1:length(dimensions)
        dim = dimensions(i);
        for x = 1:247
            for y = 1:262
                Xtemp(1:dim) = X(x,y,1:dim);
                prob = 0;
                for j = 1: C
                    prob = prob +
mvnpdf(Xtemp,mu(j,1:dim),sigma(1:dim,1:dim,j))*Pi(j);
                end
                px(x,y,i) = prob;
            end
        end
    end
    px = px.*PY;
```

```
end

function errs = countError(mask,p_fg,p_bg)
    errs = zeros(1,11);
    pred = zeros(250,270);
    for j = 1:11
        count = 0;
        for x = 1:247
            for y = 1:262
                if p_bg(x,y,j)<p_fg(x,y,j)
                    pred(x,y) = 1;
                end

                if pred(x,y) ~= mask(x,y)
                    count = count + 1;
                end
            end
        end
        errs(j) = count/(255*270);
    end

    figure
    imagesc(pred);
    colormap(gray(255));
end
```

Part b)

```
clear;
clc;
load('TrainingSamplesDCT_8_new.mat');
FG = TrainsampleDCT_FG;
BG = TrainsampleDCT_BG;

sample_BG = size(BG,1);
sample_FG = size(FG,1);

feature = size(BG,2);

img = im2double(imread('cheetah.bmp'));
mask = im2double(imread('cheetah_mask.bmp'));

%read Zig-Zag Pattern.txt file
zz = fopen('Zig-Zag Pattern.txt','r');
zzPat = fscanf(zz,'%d',[8,8])+1;
fclose(zz);

% obtain the DCT of the image
[row,colm] = size(img);
img_zzs = zeros(row-8,colm-8,64);
for i = 1:row-8
    for j = 1:colm-8
```

```
            dctImg = dct2(img(i:i+7,j:j+7));
            for x = 1:8
                for y = 1:8
                    img_zzs(i,j,zzPat(x,y)) = dctImg(x,y);
                end
            end
        end
    end
    [r,m] = size(img_zzs,1,2);
    %% Part b)
    C = [1,2,4,8,16,32];

    % Initialization
    pi_BG_1 = zeros(1,1);
    mu_BG_1 = zeros(1,feature,1);
    sigma_BG_1 = zeros(feature,feature,1,1);

    pi_FG_1 = zeros(1,1);
    mu_FG_1 = zeros(1,feature,1);
    sigma_FG_1 = zeros(feature,feature,1,1);

    pi_BG_2 = zeros(2,1);
    mu_BG_2 = zeros(2,feature,1);
    sigma_BG_2 = zeros(feature,feature,2,1);

    pi_FG_2 = zeros(1,1);
    mu_FG_2 = zeros(2,feature,1);
    sigma_FG_2 = zeros(feature,feature,2,1);

    pi_BG_4 = zeros(4,1);
    mu_BG_4 = zeros(4,feature,1);
    sigma_BG_4 = zeros(feature,feature,4,1);

    pi_FG_4 = zeros(4,1);
    mu_FG_4 = zeros(4,feature,1);
    sigma_FG_4 = zeros(feature,feature,4,1);

    pi_BG_8 = zeros(8,1);
    mu_BG_8 = zeros(8,feature,1);
    sigma_BG_8 = zeros(feature,feature,8,1);

    pi_FG_8 = zeros(8,1);
    mu_FG_8 = zeros(8,feature,1);
    sigma_FG_8 = zeros(feature,feature,8,1);

    pi_BG_16 = zeros(16,1);
    mu_BG_16 = zeros(16,feature,1);
    sigma_BG_16 = zeros(feature,feature,16,1);

    pi_FG_16 = zeros(16,1);
    mu_FG_16 = zeros(16,feature,1);
    sigma_FG_16 = zeros(feature,feature,16,1);

    pi_BG_32 = zeros(32,1);
    mu_BG_32 = zeros(32,feature,1);
```

```
sigma_BG_32 = zeros(feature,feature,32,1);

pi_FG_32 = zeros(32,1);
mu_FG_32 = zeros(32,feature,1);
sigma_FG_32 = zeros(feature,feature,32,1);

for c = C
    pi_BG = rand(1,c);
    pi_BG = pi_BG./sum(pi_BG);
    mu_BG = BG(randperm(sample_BG,c),:);

    pi_FG = rand(1,c);
    pi_FG = pi_FG./sum(pi_FG);
    mu_FG = FG(randperm(sample_FG,c),:);

    sigma_BG = zeros(feature,feature,c);
    sigma_FG = zeros(feature,feature,c);
    for i = 1:c
        sigma_BG(:,:,i) = diag(rand(1,feature)+1e-6);
        sigma_FG(:,:,i) = diag(rand(1,feature)+1e-6);
    end

    % BG EM
    [pi_BG,mu_BG,sigma_BG] = EM(c,BG,pi_BG,mu_BG,sigma_BG);
    % FG EM
    [pi_FG,mu_FG,sigma_FG] = EM(c,BG,pi_FG,mu_FG,sigma_FG);

    if c == 1
       pi_BG_1 = pi_BG;
       mu_BG_1 = mu_BG;
       sigma_BG_1 = sigma_BG;

       pi_FG_1 = pi_FG;
       mu_FG_1 = mu_FG;
       sigma_FG_1 = sigma_FG;
    elseif c == 2
       pi_BG_2 = pi_BG;
       mu_BG_2 = mu_BG;
       sigma_BG_2 = sigma_BG;

       pi_FG_2 = pi_FG;
       mu_FG_2 = mu_FG;
       sigma_FG_2 = sigma_FG;
    elseif c == 4
       pi_BG_4 = pi_BG;
       mu_BG_4 = mu_BG;
       sigma_BG_4 = sigma_BG;

       pi_FG_4 = pi_FG;
       mu_FG_4 = mu_FG;
       sigma_FG_4 = sigma_FG;
    elseif c == 8
       pi_BG_8 = pi_BG;
       mu_BG_8 = mu_BG;
       sigma_BG_8 = sigma_BG;
```

```
        pi_FG_8 = pi_FG;
        mu_FG_8 = mu_FG;
        sigma_FG_8 = sigma_FG;
    elseif c == 16
        pi_BG_16 = pi_BG;
        mu_BG_16 = mu_BG;
        sigma_BG_16 = sigma_BG;

        pi_FG_16 = pi_FG;
        mu_FG_16 = mu_FG;
        sigma_FG_16 = sigma_FG;
    else
        pi_BG_32 = pi_BG;
        mu_BG_32 = mu_BG;
        sigma_BG_32 = sigma_BG;

        pi_FG_32 = pi_FG;
        mu_FG_32 = mu_FG;
        sigma_FG_32 = sigma_FG;
    end
end
%%
% **************Calculate the Probability of FG and BG for BDR*************
dimensions = [1,2,4,8,16,24,32,40,48,56,64];
PY_BG = sample_BG/(sample_FG+sample_BG);
PY_FG = sample_FG/(sample_FG+sample_BG);

% size =  247*262*11*6
PX_BG = zeros(r,m,length(dimensions),length(C));
PX_FG = zeros(r,m,length(dimensions),length(C));

PX_BG(:,:,:,1) = calPX(1,img_zzs,pi_BG_1,mu_BG_1,sigma_BG_1,PY_BG);
PX_FG(:,:,:,1) = calPX(1,img_zzs,pi_FG_1,mu_FG_1,sigma_FG_1,PY_FG);

PX_BG(:,:,:,2) = calPX(2,img_zzs,pi_BG_2,mu_BG_2,sigma_BG_2,PY_BG);
PX_FG(:,:,:,2) = calPX(2,img_zzs,pi_FG_2,mu_FG_2,sigma_FG_2,PY_FG);

PX_BG(:,:,:,3) = calPX(4,img_zzs,pi_BG_4,mu_BG_4,sigma_BG_4,PY_BG);
PX_FG(:,:,:,3) = calPX(4,img_zzs,pi_FG_4,mu_FG_4,sigma_FG_4,PY_FG);

PX_BG(:,:,:,4) = calPX(8,img_zzs,pi_BG_8,mu_BG_8,sigma_BG_8,PY_BG);
PX_FG(:,:,:,4) = calPX(8,img_zzs,pi_FG_8,mu_FG_8,sigma_FG_8,PY_FG);

PX_BG(:,:,:,5) = calPX(16,img_zzs,pi_BG_16,mu_BG_16,sigma_BG_16,PY_BG);
PX_FG(:,:,:,5) = calPX(16,img_zzs,pi_FG_16,mu_FG_16,sigma_FG_16,PY_FG);
%%
PX_BG(:,:,:,6) = calPX(32,img_zzs,pi_BG_32,mu_BG_32,sigma_BG_32,PY_BG);
PX_FG(:,:,:,6) = calPX(32,img_zzs,pi_FG_32,mu_FG_32,sigma_FG_32,PY_FG);
%
% ************************Calculate the PoE*******************************
error_EM = zeros(6,length(dimensions));
error_EM(1,:) = countError(mask,PX_FG(:,:,:,1),PX_BG(:,:,:,1));
error_EM(2,:) = countError(mask,PX_FG(:,:,:,2),PX_BG(:,:,:,2));
error_EM(3,:) = countError(mask,PX_FG(:,:,:,3),PX_BG(:,:,:,3));
error_EM(4,:) = countError(mask,PX_FG(:,:,:,4),PX_BG(:,:,:,4));
error_EM(5,:) = countError(mask,PX_FG(:,:,:,5),PX_BG(:,:,:,5));
error_EM(6,:) = countError(mask,PX_FG(:,:,:,6),PX_BG(:,:,:,6));

figure
hold on;
plot(dimensions,error_EM(1,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(2,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(3,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(4,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(5,:),'-o','MarkerSize',3,'LineWidth',1.5);
plot(dimensions,error_EM(6,:),'-o','MarkerSize',3,'LineWidth',1.5);
legend('C=1','C=2','C=4','C=8','C=16','C=32');
title('The PoE vs. Dimension With Different Numbers of Components C');
xlabel('Dimensions');
ylabel('PoE')
grid on;
hold off;

%%
function [Pi_n,mu_n,sigma_n] = EM(C,X,Pi,mu,sigma)
    iter = 100; % maximum iteration
    likehood = zeros(1,iter); % log likehood for stopping EM
    jointpdf = zeros(size(X,1),C);
    mu_n = zeros(C,size(X,2));
    sigma_n = zeros(size(X,2),size(X,2),C);
    % EM
    for h = 1:iter
        % check the log likehood to stop the function once meet the
        % condition
        if h > 1
            if(abs((likehood(h)-likehood(h-1))/likehood(h))<0.01)
                break;
            end
        end

        % E-step
        for i = 1:C
            for j = 1:size(X,1)
                jointpdf(j,i) = sqrt((2*pi)^64*det(sigma(:,:,i)))^(-1)*exp(-(X(j,:)-
mu(i,:))/sigma(:,:,i)*(X(j,:)-mu(i,:))'/2)*Pi(i);
            end
        end
        hij = jointpdf ./ sum(jointpdf,2);

        % M-step
        Pi_n = sum(hij,1)/size(X,1);
        for i = 1:C
            mu_temp = 0;
            sigma_temp = 0;
            for j = 1:size(X,1)
                mu_temp = mu_temp + hij(j,i)*X(j,:);
                sig_temp = diag((X(j,:)-mu(i,:))'*(X(j,:)-mu(i,:)));
                sig_temp(sig_temp<1e-5) = 1e-5;
                sigma_temp = sigma_temp + hij(j,i)*diag(sig_temp);
            end
            mu_n(i,:) = mu_temp/sum(hij(:,i),1);
            sigma_n(:,:,i) = sigma_temp/sum(hij(:,i),1);
        end
    end
end

function px = calPX(C,X,Pi,mu,sigma,PY)
    px = zeros(247,262,11);
    dimensions = [1,2,4,8,16,24,32,40,48,56,64];
    for i = 1:length(dimensions)
        dim = dimensions(i);
        for x = 1:247
            for y = 1:262
                Xtemp(1:dim) = X(x,y,1:dim);
                prob = 0;
                for j = 1: C
                    prob = prob +
mvnpdf(Xtemp,mu(j,1:dim),sigma(1:dim,1:dim,j))*Pi(j);
                end
                px(x,y,i) = prob;
            end
        end
    end
    px = px.*PY;
end

function errs = countError(mask,p_fg,p_bg)
    pred = zeros(255,270);
    errs = zeros(1,11);
    for j = 1:11
        count = 0;
        for x = 1:247
            for y = 1:262
                if p_bg(x,y,j)<=p_fg(x,y,j)
                    pred(x,y) = 1;
                end

                if pred(x,y) ~= mask(x,y)
                    count = count + 1;
                end
            end
        end
        errs(j) = count/(255*270);
    end
%
    figure
    imagesc(pred);
    colormap(gray(255));
end
```