Assignment #5

1 1.Computation Equation

The EM update equation we use is as following:

$$h_{ij} = \frac{G(x_i, \mu_j^{(n)}, \sigma_j^{(n)})\pi_j^{(n)}}{\sum_{k=1}^{C} G(x_i, \mu_k^{(n)}, \sigma_k^{(n)})\pi_k^{(n)}}$$

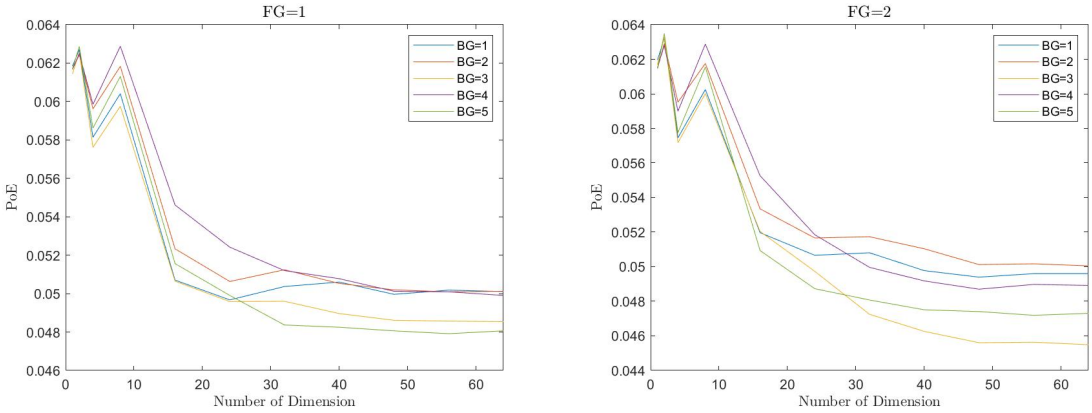
$$\mu_j^{(n+1)} = \frac{\sum_i h_{ij} x_i}{\sum_i h_{ij}}$$

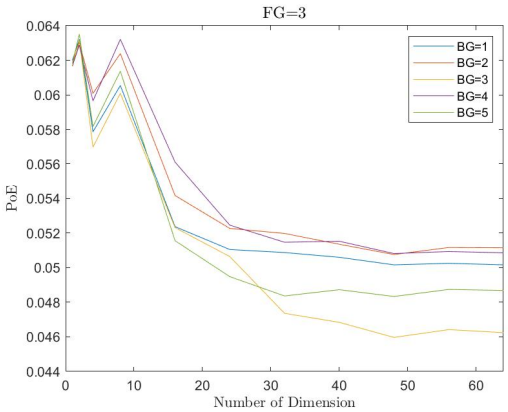
$$\pi_j^{(n+1)} = \frac{1}{n}\sum_i h_{ij}$$

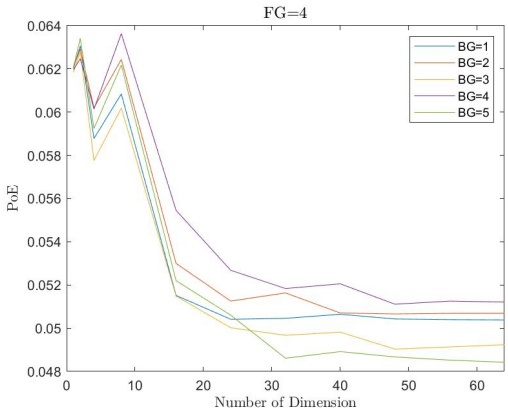
$$\sigma_j^{2(n+1)} = \frac{\sum_i h_{ij}(x_i - \mu_j)^2}{\sum_i h_{ij}}$$

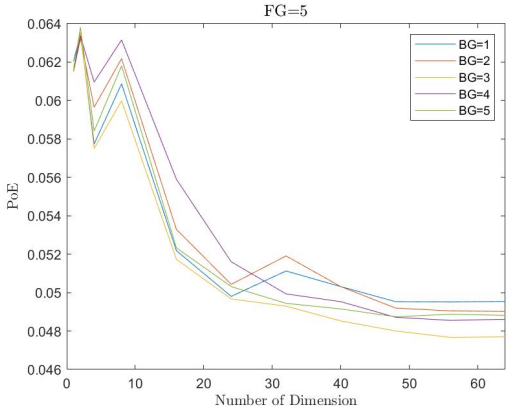
2 Problem (a)

The following is the plot of PoE versus dimension for each of 25 classifiers obtained with all possible mixture pairs. FG represents the mixture of Gaussian to cheetah class and BG represents the mixture of Gaussian to grass class.



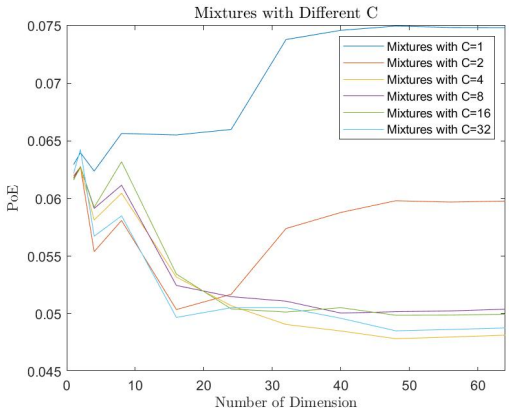






3 Problem (b)

The following is the PoE versus dimension for each number of mixture components.



Appendix

The following is the Matlab code.

3.1 HW5_solution.m

```
%%
clear;
clc;
tic;
%%
%Training
%Read the TrainingSamplesDCT_8.mat file
load('dataset/TrainingSamplesDCT_8_new.mat');
%Save TrainsampleDCT_BG and TrainsampleDCT_FG in temporary value
train_BG = TrainsampleDCT_BG;
train_FG = TrainsampleDCT_FG;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compute parameter from EM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C = 8;
for M=1:5
    fun_EM(C, train_BG, train_FG, M);
end
for C=[1,2,4,8,16,32]
    fun_EM(C, train_BG, train_FG, 0);
end
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BDR with different mixture pair and
% dimension
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
%Read the TrainingSamplesDCT_8.mat file
load('dataset/TrainingSamplesDCT_8_new.mat');
%Save TrainsampleDCT_BG and TrainsampleDCT_FG in temporary value
train_BG = TrainsampleDCT_BG;
train_FG = TrainsampleDCT_FG;
%Compute threshold
P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
T = P_BG / P_FG;
for FG=1:5
    load(['savedata/MuSigma_C=8M=',num2str(FG),'.mat'],'mu_FG', 'sigma_FG'
    , 'weight_FG');
    for BG=1:5
        disp(['Start with FG=',num2str(FG),',BG=',num2str(BG)]);
```

```
        load(['savedata/MuSigma_C=8M=',num2str(BG),'.mat'],'mu_BG', '
    sigma_BG', 'weight_BG');
        error = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG,
    sigma_FG);
        save(['savedata/error_FG=',num2str(FG),'&BG=',num2str(BG),'.mat'],
    'error');
    end
end
for C=[1,2,4,8,16,32]
    disp(['Start with C=',num2str(C)]);
    load(['savedata/MuSigma_C=',num2str(C),'M=',num2str(0),'.mat'],'mu_BG'
    ,...
        'mu_FG','sigma_BG','sigma_FG','weight_BG','weight_FG');
    error = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG,
    sigma_FG);
    save(['savedata/error_C=',num2str(C),'&M=',num2str(0),'.mat'],'error')
    ;
end

toc;
```

3.2 HW5_plot.m

```
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot for problem a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dimension = [1,2,4,8,16,24,32,40,48,56,64];
for FG=1:1:5
    for BG=1:1:5
        load(['savedata/error_FG=',num2str(FG),'&BG=',num2str(BG),'.mat'])
    ;
        %Plot the data
        figure(1);
        plot(dimension,error,'DisplayName',['BG=',num2str(BG)]);
        hold on;
    end
    legend
    xlim([0 64]);
    title({['FG=',num2str(FG)]},'Fontsize',12,'interpreter','latex');
    ylabel('PoE', 'interpreter', 'latex');
    xlabel('Number of Dimension', 'interpreter', 'latex');
    saveas(gcf, ['images/FG=',num2str(FG)','.jpg']);
    close(gcf);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot for problem a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dimension = [1,2,4,8,16,24,32,40,48,56,64];
```

```
for C=[1,2,4,8,16,32]
    load(['savedata/error_C=',num2str(C),'&M=',num2str(0),'.mat']);
    %Plot the data
    figure(1);
    plot(dimension,error,'DisplayName',['Mixtures with C=',num2str(C)]);
    hold on;
end
legend
xlim([0 64]);
title({['Mixtures with Different C']},'Fontsize',12,'interpreter','latex')
   ;
ylabel('PoE', 'interpreter', 'latex');
xlabel('Number of Dimension', 'interpreter', 'latex');
saveas(gcf, ['images/mix_C.jpg']);
close(gcf);
```

3.3 fun_getinit.m

This function performs parameters initialization.

```
function [weight, mu, sigma] = fun_getinit(C,data)
% This function is parameter initialization
% C -- number of mixtures we want to use
%
% Return:
% weight -- the matrix of weight, size of (C,1)
% mu -- the matrix of mean, size of (C, 64)
% sigma -- tensor of sigma, size of (C, 64)

% Initialize weight
weight = rand(C,1);
weight = weight / sum(weight);
% Initialize mu
mu = 0.1 * rand(C,64) + mean(data);
% Initialize sigma
sigma = rand(C, 64);

end
```

3.4 fun_BDR.m

This function performs BDR classification using parameters we learn from EM.

```
function [error] = fun_BDR(T, weight_BG, weight_FG, mu_BG, mu_FG, sigma_BG
   , sigma_FG)
% This function is for BDR
% weight -- the matrix of weight, size of (C,1)
% mu -- the matrix of mean, size of (C, 64)
% sigma -- tensor of sigma, size of (C, 64)
```

```
6
7  %Load DCT file
8  load('DCT_coeffience.mat');
9  dimension = [1,2,4,8,16,24,32,40,48,56,64];
10 C = length(weight_BG);
11 error = [];
12 for d=dimension
13     DCT_coeffience_d = DCT_coeffience(:, 1:d);
14     % Select the corresponding parameters
15     mu_FG_d = mu_FG(:, 1:d);
16     mu_BG_d = mu_BG(:, 1:d);
17     sigma_FG_d = reshape(repmat(eye(d), [1, C])*diag(reshape(sigma_FG(:,
       1:d)', [C*d, 1])), [d, d, C]);
18     sigma_BG_d = reshape(repmat(eye(d), [1, C])*diag(reshape(sigma_BG(:,
       1:d)', [C*d, 1])), [d, d, C]);
19     % Compute a predict mask
20     MixG_FG = gmdistribution(mu_FG_d,sigma_FG_d,weight_FG);
21     MixG_BG = gmdistribution(mu_BG_d,sigma_BG_d,weight_BG);
22     I_pre = pdf(MixG_FG, DCT_coeffience_d) > T * pdf(MixG_BG,
       DCT_coeffience_d);
23     % Compute PoE
24     error_temp = fun_error(I_pre);
25     error = [error,error_temp];
26 end
27
28 end
```

3.5 fun_EM.m

This function performs EM.

```
1  function [] = fun_EM(C, data_BG, data_FG, M)
2  % This function is for EM
3  % C -- number of mixtures we want to use
4  % M -- number of mixture pair, used as a lable
5  % data_BG -- training data of BG
6  % data_FG -- training data of FG
7  %
8  % The following is abstract process:
9  % 1. Initialize parameters of mixture gaussian and weight
10 % 2. Compute initial h matrix
11 % 3. Loop of EM
12
13 loop = 1200;
14 % For BG EM
15 disp(['Start BG EM with C=',num2str(C),',M=',num2str(M)]);
16 N = size(data_BG,1); % number of samples
17 D = 64; % number of dimension
18 h_BG = zeros(N,C); % define H matrix
```

```
[weight_BG, mu_BG, sigma_BG] = fun_getinit(C,data_BG);
% weight -- the matrix of weight, size of (C,1)
% mu -- the matrix of mean, size of (C, 64)
% sigma -- tensor of sigma, size of (C, 64)
% h_BG -- size of (N, C)
% data_BG -- size of (N, 64)
weight_BG_pre = weight_BG;
mu_BG_pre = mu_BG;
sigma_BG_pre = sigma_BG;

for step=1:loop
    % Compute h_BG matrix
    for j=1:C
        h_BG(:,j) = weight_BG_pre(j)*mvnpdf(data_BG, mu_BG_pre(j,:), diag(
   sigma_BG_pre(j,:)));
    end
    h_BG = h_BG ./ sum(h_BG, 2);
    % Update parameter
    % Update weight
    weight_BG = mean(h_BG, 1)';
    % Update Sigma
    sigma_temp = sum(reshape(h_BG, [1,N,C]).*(data_BG'-reshape(mu_BG_pre',
    [D,1,C])).^2, 2);
    sigma_BG = squeeze(sigma_temp)' ./ sum(h_BG, 1)';
    sigma_BG(sigma_BG < 1e-4) = 1e-4;
    % Update mu
    mu_BG = h_BG'*data_BG ./ sum(h_BG, 1)';
%     if max(max(abs(mu_BG - mu_BG_pre)./abs(mu_BG_pre))) < 0.001
%         break;
%     end
    weight_BG_pre = weight_BG;
    mu_BG_pre = mu_BG;
    sigma_BG_pre = sigma_BG;
end


% For FG EM
disp(['Start FG EM with C=',num2str(C)]);
N = size(data_FG,1); % number of samples
h_FG = zeros(N,C);
[weight_FG, mu_FG, sigma_FG] = fun_getinit(C,data_FG);
% weight -- the matrix of weight, size of (C,1)
% mu -- the matrix of mean, size of (C, 64)
% sigma -- tensor of sigma, size of (64, 64, C)
% h_BG -- size of (N,C)
% data_BG -- size of (N,64)
weight_FG_pre = weight_FG;
mu_FG_pre = mu_FG;
sigma_FG_pre = sigma_FG;

```

```
for step=1:loop
    % Compute h_FG matrix
    for j=1:C
        h_FG(:,j) = weight_FG_pre(j)*mvnpdf(data_FG, mu_FG_pre(j,:), diag(
   sigma_FG_pre(j,:)));
    end
    h_FG = h_FG ./ sum(h_FG, 2);
    % Update parameter
    % Update weight
    weight_FG = mean(h_FG, 1)';
    % Update sigma
    sigma_temp = sum(reshape(h_FG, [1,N,C]).*(data_FG'-reshape(mu_FG_pre',
    [D,1,C])).^2, 2);
    sigma_FG = squeeze(sigma_temp)' ./ sum(h_FG, 1)';
    sigma_FG(sigma_FG < 1e-4) = 1e-4;
    % Update mu
    mu_FG = h_FG'*data_FG ./ sum(h_FG, 1)';
%     if max(max(abs(mu_FG - mu_FG_pre)./abs(mu_FG_pre))) < 0.001
%         break;
%     end
    weight_FG_pre = weight_FG;
    mu_FG_pre = mu_FG;
    sigma_FG_pre = sigma_FG;
end

save(['savedata/MuSigma_C=',num2str(C),'M=',num2str(M),'.mat'],'mu_BG',...
    'mu_FG','sigma_BG','sigma_FG','weight_BG','weight_FG')
end
```

3.6 fun_error.m

This function performs error computation.

```
function [result] = fun_error(I_pre)
% This function is for computing error

%Read the mask file
I = imread('dataset/cheetah_mask.bmp');
I = im2double(I);
%Define the predict mask
mask_64 = zeros(size(I));
%Load DCT file
load('DCT_coeffience.mat');
%Define the loop numbers
loop_row = size(I,1) - 8 + 1;
loop_column = size(I,2) - 8 + 1;
k=1;
for i=1:1:loop_row
    for j=1:1:loop_column
```

```
        mask_64(i,j) = I_pre(k);
        k=k+1;
    end
end
%Calculate the probability of error
result = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
end
```