Homework Set Three
ECE 271A

Name : Hsiu-Wen Yen

ID : A59010599

a) To compute **Bayesian BDR**

$$\boxed{\begin{aligned} &i^*(x) = \arg\max_i P_{X|Y,T}(x \mid i, D_i) P_Y(i) \\ &where \quad P_{X|Y,T}(x \mid i, D_i) = \int P_{X|Y,\Theta}(x \mid i, \theta) P_{\Theta|Y,T}(\theta \mid i, D_i) d\theta \end{aligned}}$$

We need to calculate $P_{X|Y,T}(x|i, D_i)$, which is $P_{X|T}(x|D_1)$ in problem a).

First, we compute $P_{\Theta|i,T}(\theta|i, D_i)$, which is $P_{\mu|T}(\mu|D_1)$.
From previous lecture, we can derive $P_{\mu|T}$ by this formula.

$$\boldsymbol{P}_{\mu|\boldsymbol{T}}(\mu \mid \boldsymbol{D}) \propto \prod_{\boldsymbol{i}} \boldsymbol{P}_{\boldsymbol{X}|\mu}(\boldsymbol{x}_{\boldsymbol{i}} \mid \mu) \boldsymbol{P}_{\mu}(\mu)$$

$$\propto \prod_{\boldsymbol{i}} \boldsymbol{G}(\boldsymbol{x}_{\boldsymbol{i}}, \mu, \sigma^2) \boldsymbol{G}(\mu, \mu_0, \sigma_0^2)$$

Therefore, $P_{\mu|T}$ can be written as

$$\boxed{P_{\mu|T}(\mu \mid D) = G\left(\mu, \mu_n, \sigma_n^2\right)}$$

$$\mu_n = \frac{\sigma_0^2 \sum_i x_i + \mu_0 \sigma^2}{\sigma^2 + n\sigma_0^2} \Rightarrow \boxed{\mu_n = \underbrace{\frac{n\sigma_0^2}{\sigma^2 + n\sigma_0^2}}_{\alpha_n} \mu_{ML} + \underbrace{\frac{\sigma^2}{\sigma^2 + n\sigma_0^2}}_{1-\alpha_n} \mu_0}$$

$$\sigma_n^2 = \left(\frac{\sigma^2 \sigma_0^2}{\sigma^2 + n\sigma_0^2}\right) \Rightarrow \boxed{\frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}$$

In our 64-dimesion case, it is

$$\mu_n = \frac{n\Sigma_0}{\Sigma + n\Sigma_0} \mu_{ML} + \frac{\Sigma}{\Sigma + n\Sigma_0} \mu_0$$

$$\Sigma_n = \frac{\Sigma \Sigma_0}{\Sigma + n\Sigma_0}$$

μ_0 is given, which can be from strategy 1 or 2.
Σ_0 is given, it's same for strategy 1 and 2.
μ_{ML} can be computed from dataset.
Σ can be computed from dataset.

Next, we can calculate predictive distribution by

$$\boxed{P_{X|T}(x \mid D) = G(x, \mu_n, \sigma^2 + \sigma_n^2)}$$

In our 64-dimesion case, it is

$$P_{X|T}(x|D) = G(x, \mu_n, \Sigma + \Sigma_n)$$

Plug this into BDR to do classification.

b) To compute **ML BDR**

$$\boxed{\begin{aligned} i^*(x) &= \arg\max_i P_{X|Y}\left(x \mid i;\theta_i^*\right)P_Y(i) \\ \text{where } \theta_i^* &= \arg\max_\theta P_{X|Y}\left(D \mid i,\theta\right) \end{aligned}}$$

It's exactly same as what we did in HW2.

$\mu^* = \frac{1}{n}\Sigma x_i = \mu_{ML}$ in a)

$\Sigma^* = \frac{1}{n}\Sigma(x_i - \mu *)(x_i - \mu *)^T = \Sigma$ in a)

Therefore,

$$P_{X|Y}(x|i) = G(x, \mu_{ML}, \Sigma)$$

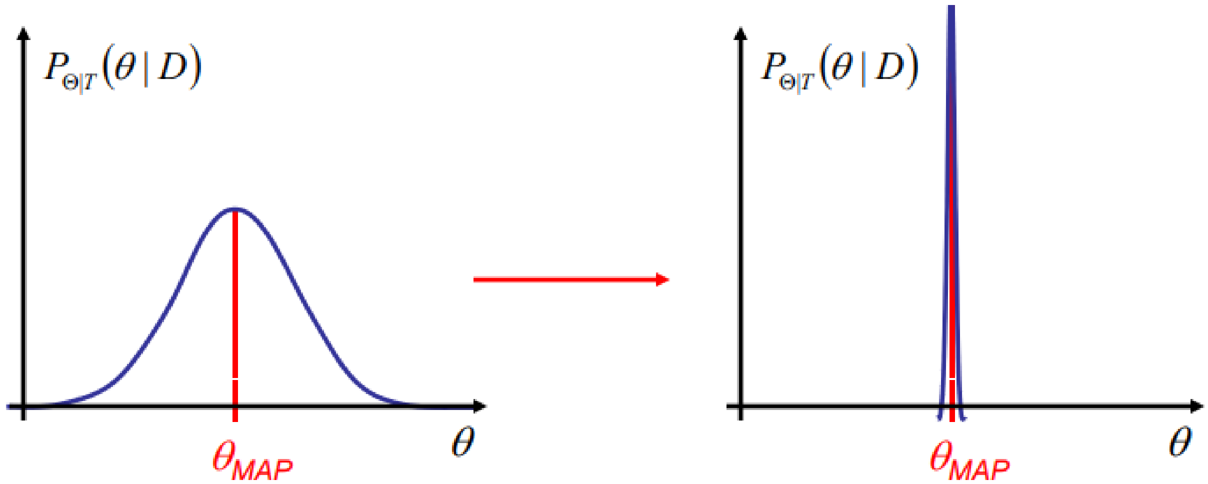
Plug this into BDR to do classification.

c) Using **MAP approximation** to compute **Bayes MAP-BDR**

We can use delta function to convert our posterior probability into

$$\boxed{\theta_{MAP} = \arg\max_\theta P_{\Theta|T}\left(\theta \mid D\right)}$$

Illustrate it as picture,



Therefore, we will get

$$\boxed{\begin{aligned} P_{X|T}(x \mid D) &= \int P_{X|\Theta}(x \mid \theta)\delta(\theta - \theta_{MAP})d\theta \\ &= P_{X|\Theta}(x \mid \theta_{MAP}) \end{aligned}}$$

It is same as what problem c) states.

$$P_{\mathbf{x}|\mathbf{T}}(\mathbf{x}|\mathcal{D}_1) = P_{\mathbf{x}|\mu}(\mathbf{x}|\mu_{MAP})$$

$$\mu_{MAP} = \arg\max_{\mu} P_{\mu|\mathbf{T}}(\mu|\mathcal{D}_1)$$

From problem a), we knew $P_{\mu|T}(\mu|D_1) = G(x, \mu_n, \Sigma_n)$, which means $\mu_{MAP} = \mu_n$.

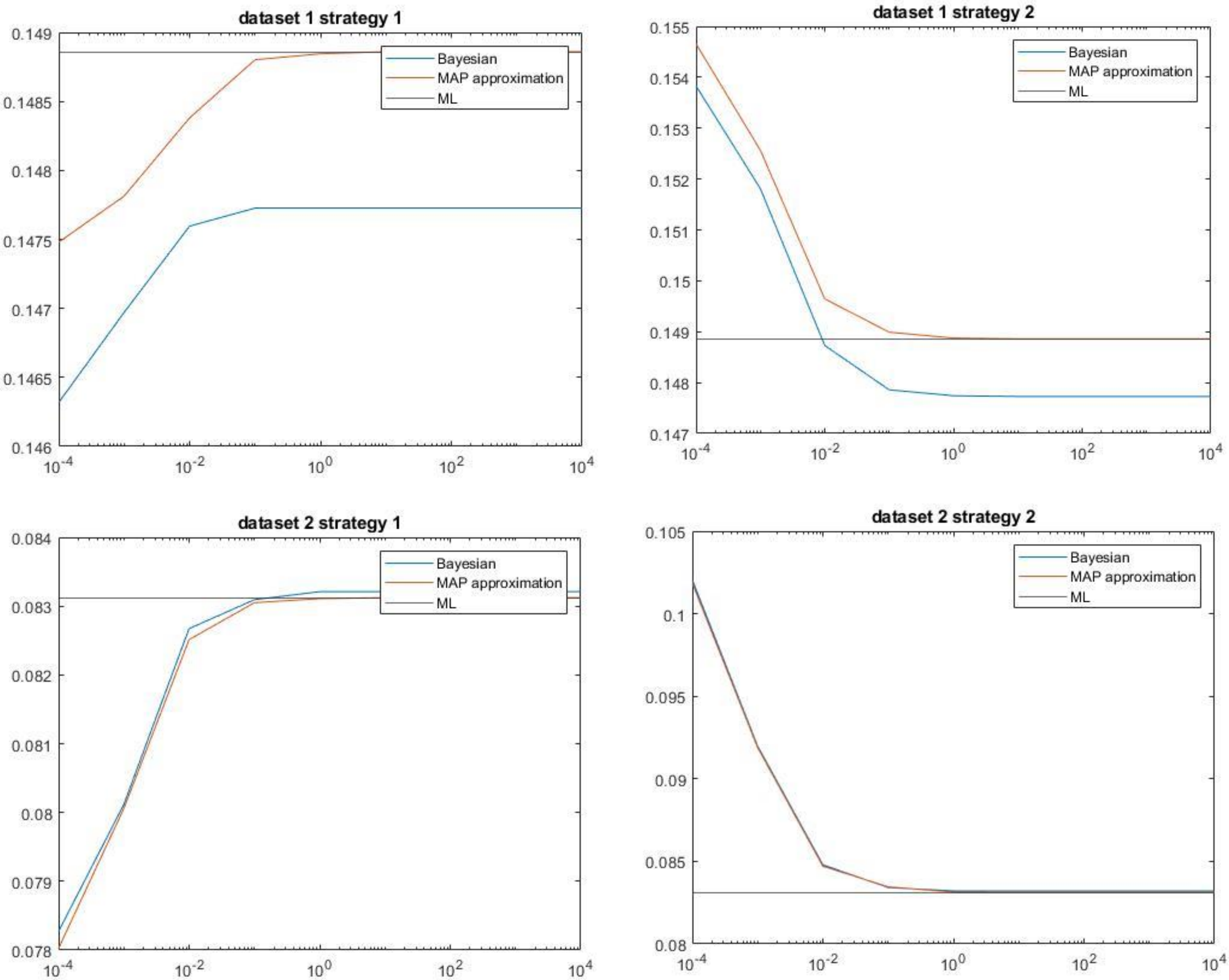
In the end, we get

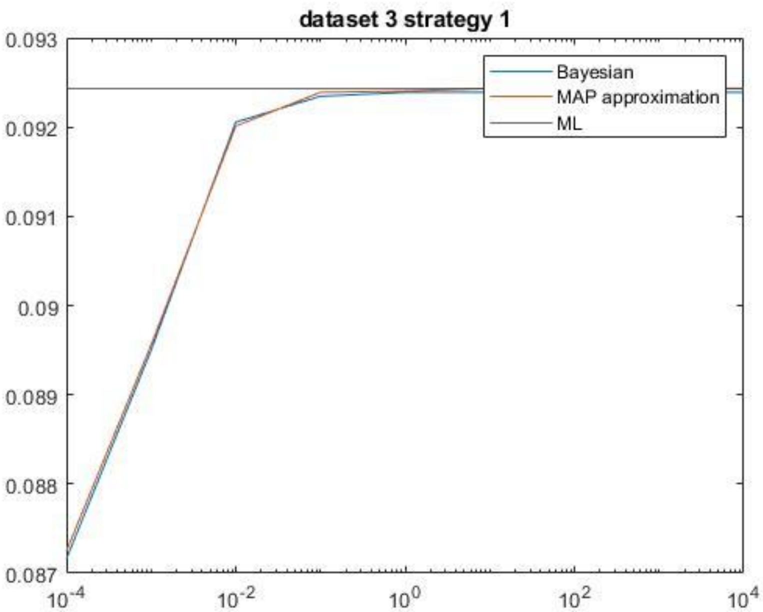
$$P_{X|T}(x|D) = G(x, \mu_n, \Sigma)$$

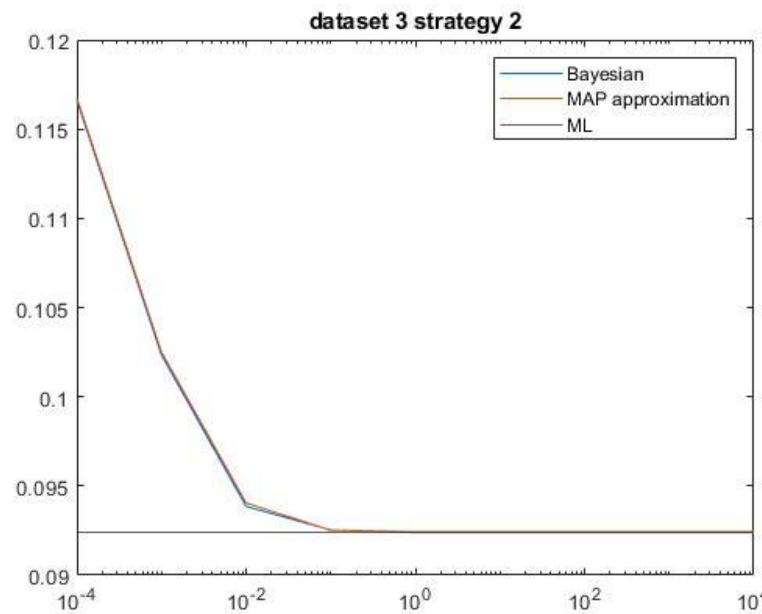
Plug this into BDR to do classification.

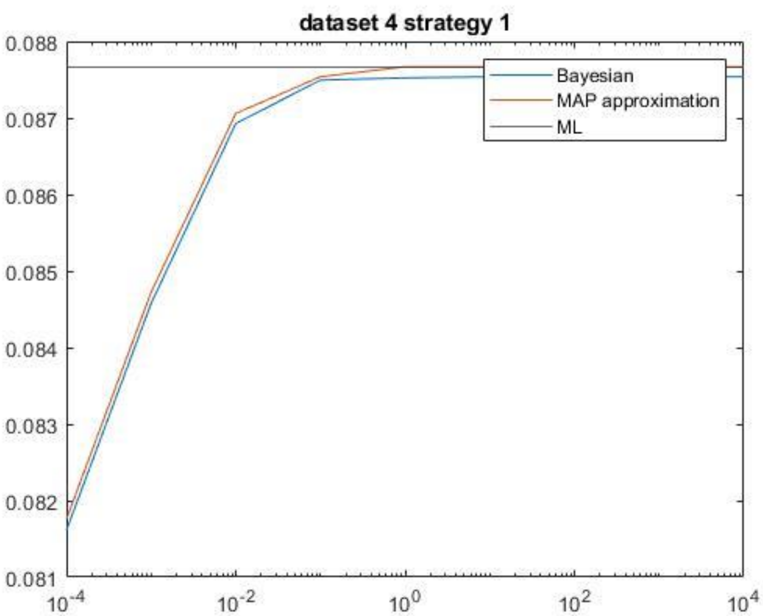
d) e) Plot 4 Dataset with 2 strategies.

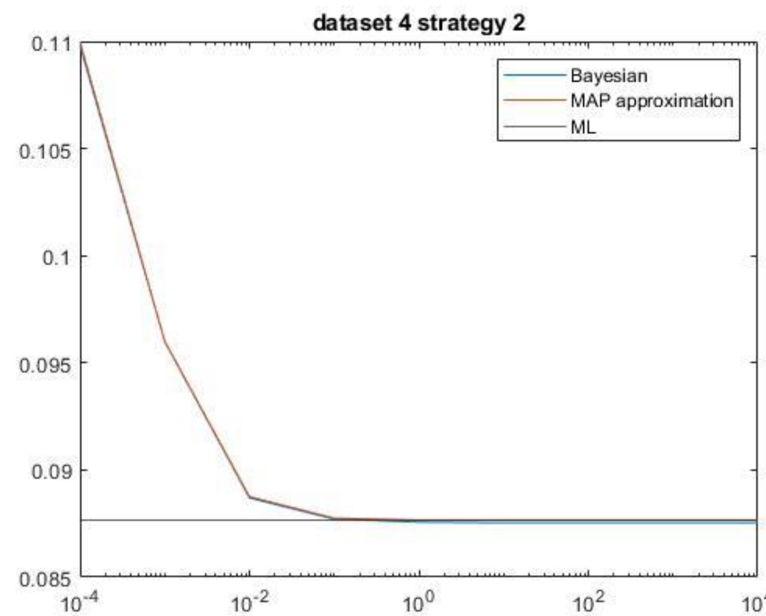
Result:




dataset 3 strategy 1
Bayesian
MAP approximation
ML
0.093
0.092
0.091
0.09
0.089
0.088
0.087
10^-4
10^-2
10^0
10^2
10^4


dataset 3 strategy 2
Bayesian
MAP approximation
ML
0.12
0.115
0.11
0.105
0.1
0.095
0.09
10^-4
10^-2
10^0
10^2
10^4


dataset 4 strategy 1
Bayesian
MAP approximation
ML
0.088
0.087
0.086
0.085
0.084
0.083
0.082
0.081
10^-4
10^-2
10^0
10^2
10^4


dataset 4 strategy 2
Bayesian
MAP approximation
ML
0.11
0.105
0.1
0.095
0.09
0.085
10^-4
10^-2
10^0
10^2
10^4

1. explain the relative behavior of these three curves

It's clear to see when α is small, gap between three curves is large, and gap becomes small when α increase.

This is because α decides our Σ_0. According $\mu_n = \frac{n\Sigma_0}{\Sigma+n\Sigma_0}\mu_{ML} + \frac{\Sigma}{\Sigma+n\Sigma_0}\mu_0$

when Σ_0 is bigger, weighting for μ_{ML} increases, and weighting for μ_0 decreases.

Therefore, when α is big enough, μ_n for Bayesian BDR and Bayes MAP-BDR will close to μ_{ML}.

Since Bayes MAP-BDR and ML-BDR share same covariance matrix, so their probability of error is very close when α is big enough.

When α is small, weighting for μ_{ML} decreases, and weighting for μ_0 increases. It means $\mu_{ML} \neq \mu_n$. Therefore, probability of error for MAP-BDR and ML-BDR is very different even their covariance matrixes are same.

2. how that behavior changes from dataset to dataset

The difference in each dataset is the number of samples.

When the number of data increases, it is clear to see the difference between three curves decrease significantly.

The reason for this can be derived from formula we used before.

From $\Sigma_n = \frac{\Sigma\Sigma_0}{\Sigma+n\Sigma_0}$, it's clear to see when α is big enough (it makes Σ_0 large) that $\Sigma_n = \frac{\Sigma\Sigma_0}{n\Sigma_0} = \frac{\Sigma}{n}$. Therefore, when n is large, it makes Σ_n close to zero. Since covariance matrix for Bayesian BDR is $\Sigma + \Sigma_n$, we can get $P_{X|T}(x|D) = G(x, \mu_n, \Sigma)$ when n is large. $G(x, \mu_n, \Sigma)$ is exactly how we compute MAP-BDR.

It concludes that the plot of probability of error for Bayesian BDR and MAP-BDR is similar when n is large (α also needs to be large enough).

3. how all of the above change when strategy 1 is replaced by strategy 2

In strategy 1 probability of error increases as α increases, and in strategy 2 probability of error decreases as α increases. This is because these two strategies using different μ_0 for prior. Before computation, we can guess strategy 1 will perform better than strategy 2 because assigning different μ_0 for these two classes (1 for lighter pattern and 3 for darker pattern) is more reasonable than using same μ_0 for different classes.

The result is exactly as what we guess. When α is small, we rely on prior more to compute μ_n.

We can see what happens when α is small. In strategy 1, probability of error for Bayesian BDR is less than ML-BDR. However, probability of error for Bayesian BDR is greater than ML-BDR in strategy 2.

When α is large, it doesn't matter we use strategy 1 or 2. Since μ_n will converge to μ_{ML}, the probability of error for all method tends to be equal.

It testifies that our assumption for μ_0 in strategy 1 is better than strategy 2.

Code Review

```
clear; clc;
load('Prior_1.mat');
load('Alpha.mat');
load('TrainingSamplesDCT_subsets_8.mat');
cheetah = imread('cheetah.bmp');
cheetah=double(cheetah)/255;
% store data into list
Data_FG_list = {D1_FG, D2_FG, D3_FG, D4_FG};
Data_BG_list = {D1_BG, D2_BG, D3_BG, D4_BG};


for strategy_idx = 1:2

if (strategy_idx == 1)
    load('Prior_1.mat');
else
    load('Prior_2.mat');
end

for data_index = 1:4

Data_FG = cell2mat(Data_FG_list(data_index));
Data_BG = cell2mat(Data_BG_list(data_index));

% ML mu for cheetah nad grass
D1_FG_mean = mean(Data_FG);
D1_BG_mean = mean(Data_BG);

% Covariance for FG and BG
D1_FG_covariance = cov(Data_FG) * ((length(Data_FG)-1)) / (length(Data_FG));
D1_BG_covariance = cov(Data_BG) * ((length(Data_BG)-1)) / (length(Data_BG));

% here is for problem a)
% loop start
for ii = 1:9

% Cov0
```

```
cov0 = diag(alpha(ii) * W0);

% cheetah_mu_n
a1_FG = length(Data_FG) * cov0 / (D1_FG_covariance + length(Data_FG)*cov0);
a2_FG = D1_FG_covariance / (D1_FG_covariance + length(Data_FG) * cov0);
cheetah_mu_n = (a1_FG * D1_FG_mean.' + a2_FG * mu0_FG.').';

% grass_mu_n
a1_BG = length(Data_BG) * cov0 / (D1_BG_covariance + length(Data_BG)*cov0);
a2_BG = D1_BG_covariance / (D1_BG_covariance + length(Data_BG) * cov0);
grass_mu_n = (  a1_BG * D1_BG_mean.' + a2_BG * mu0_BG.').';

% cheetah_covariance_n
cheetah_covariance_n = D1_FG_covariance * cov0 / (D1_FG_covariance + length(Data_FG)
* cov0);
% grass_covariance_n
grass_covariance_n = D1_BG_covariance * cov0 / (D1_BG_covariance + length(Data_BG) *
cov0);

% posteria_cheetah_covariance
posteria_cheetah_covariance = cheetah_covariance_n + D1_FG_covariance;
% posteria_grass_covariance
posteria_grass_covariance = grass_covariance_n + D1_BG_covariance;

% prior probability for class
p_cheetah = size(Data_FG,1) / (size(Data_FG,1) + size(Data_BG,1));
p_grass = size(Data_BG,1) / (size(Data_FG,1) + size(Data_BG,1));

% start to classify
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A = 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_value = expand_zigzag(block);
        % calculate P(1,x) and P(0,x), find bigger one

        P_0 = (-0.5*(x_value - grass_mu_n)/ posteria_grass_covariance * (x_value -
grass_mu_n).') - log(sqrt(det(posteria_grass_covariance)*(2*pi)^64)) + log(p_grass);

        P_1 = (-0.5*(x_value - cheetah_mu_n)/ posteria_cheetah_covariance * (x_value
- cheetah_mu_n).') - log(sqrt(det(posteria_cheetah_covariance)*(2*pi)^64)) +
log(p_cheetah);
        if (P_0 >= P_1)
            A(rows, columns) = 0;
        else
            A(rows, columns) = 1;
        end
    end
end
```

```
% calculate error
% load cheetah mask.bmp
truth = imread("cheetah_mask.bmp");
truth = double(truth/255);
err = truth - A;
err = abs(err);
probability_error = sum(err,'all') / (size(A,1)*size(A,2));

storage(ii) = probability_error;
end
% loop end

% here is for problem b)
% start to classify for ML
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A = 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_value = expand_zigzag(block);
        % calculate P(1|x) and P(0|x), find bigger one

        P_0 = (-0.5*(x_value - D1_BG_mean)/ D1_BG_covariance * (x_value -
D1_BG_mean).') - log(sqrt(det(D1_BG_covariance)*(2*pi)^64)) + log(p_grass);

        P_1 = (-0.5*(x_value - D1_FG_mean)/ D1_FG_covariance * (x_value -
D1_FG_mean).') - log(sqrt(det(D1_FG_covariance)*(2*pi)^64)) + log(p_cheetah);
        if (P_0 >= P_1)
            A(rows, columns) = 0;
        else
            A(rows, columns) = 1;
        end
    end
end
err = truth - A;
err = abs(err);
probability_error = sum(err,'all') / (size(A,1)*size(A,2));



% here is for problem c)
% loop start
for ii = 1:9

% Cov0
cov0 = diag(alpha(ii) * W0);

% cheetah_mu_n
a1_FG = cov0 / (cov0 + (1/length(Data_FG))* D1_FG_covariance);
a2_FG = (1/length(Data_FG)) * D1_FG_covariance / (cov0 + (1/length(Data_FG))*
D1_FG_covariance);
```

```
cheetah_mu_n = (a1_FG * D1_FG_mean.' + a2_FG * mu0_FG.').';

% grass_mu_n
a1_BG = cov0 / (cov0 + (1/length(Data_BG))* D1_BG_covariance);
a2_BG = (1/length(Data_BG)) * D1_BG_covariance / (cov0 + (1/length(Data_BG))*
D1_BG_covariance);
grass_mu_n = (a1_BG * D1_BG_mean.' + a2_BG * mu0_BG.').';

% cheetah_covariance_n
cheetah_covariance_n = cov0 / (cov0 + (1/length(Data_FG))* D1_FG_covariance) *
((1/length(Data_FG))*D1_FG_covariance);
% grass_covariance_n
grass_covariance_n = cov0 / (cov0 + (1/length(Data_BG))* D1_BG_covariance) *
((1/length(Data_BG))*D1_BG_covariance);

% posteria_cheetah_covariance
posteria_cheetah_covariance = cheetah_covariance_n + D1_FG_covariance;
% posteria_grass_covariance
posteria_grass_covariance = grass_covariance_n + D1_BG_covariance;

p_cheetah = size(Data_FG,1) / (size(Data_FG,1) + size(Data_BG,1));
p_grass = size(Data_BG,1) / (size(Data_FG,1) + size(Data_BG,1));

% start to classify
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A = 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_value = expand_zigzag(block);
        % calculate P(1|x) and P(0|x), find bigger one

        P_0 = (-0.5*(x_value - grass_mu_n)/ D1_BG_covariance * (x_value -
grass_mu_n).') - log(sqrt(det(D1_BG_covariance)*(2*pi)^64)) + log(p_grass);

        P_1 = (-0.5*(x_value - cheetah_mu_n)/ D1_FG_covariance * (x_value -
cheetah_mu_n).') - log(sqrt(det(D1_FG_covariance)*(2*pi)^64)) + log(p_cheetah);
        if (P_0 >= P_1)
            A(rows, columns) = 0;
        else
            A(rows, columns) = 1;
        end
    end
end
% error
% load cheetah mask.bmp
truth = imread("cheetah_mask.bmp");
truth = double(truth/255);

err = truth - A;
err = abs(err);
```

```
probability_error = sum(err,'all') / (size(A,1)*size(A,2));
storage2(ii) = probability_error;
% loop end
end

%  plot error
figure();
semilogx(alpha,storage);
hold on;
semilogx(alpha,storage2);
yline(probability_error);
legend('Bayesian','MAP approximation','ML')
txt = "dataset " + int2str(data_index) +" strategy " + int2str(strategy_idx);
title(txt);
hold off;
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
```