ECE 271A: Statistical Learning I

Homework 2

Shang-Wei Hung
A53267981

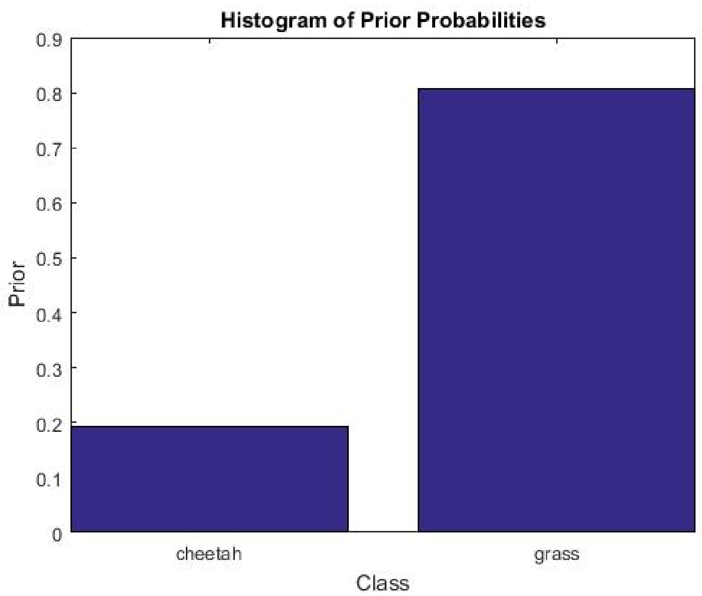
Problem 6

(a)

According to the indication of the ML estimate of the parameters of a multinomial distribution in Problem 2, prior probabilities overall n independent observations form random variable X such that $P_X(k) = \pi_k, k \in \{cheetah, grass\}$ can be expressed as $P_X(i) = \frac{c_i}{n}$, where c_i means the number of observations of each class. Thus, the prior probabilities can computed as below and the histogram estimate of the prior probabilities is shown in Figure 1.

$$P_Y(cheetach) = \frac{250}{250 + 1053} = 0.1919$$

$$P_Y(grass) = \frac{1053}{250 + 1053} = 0.8081$$

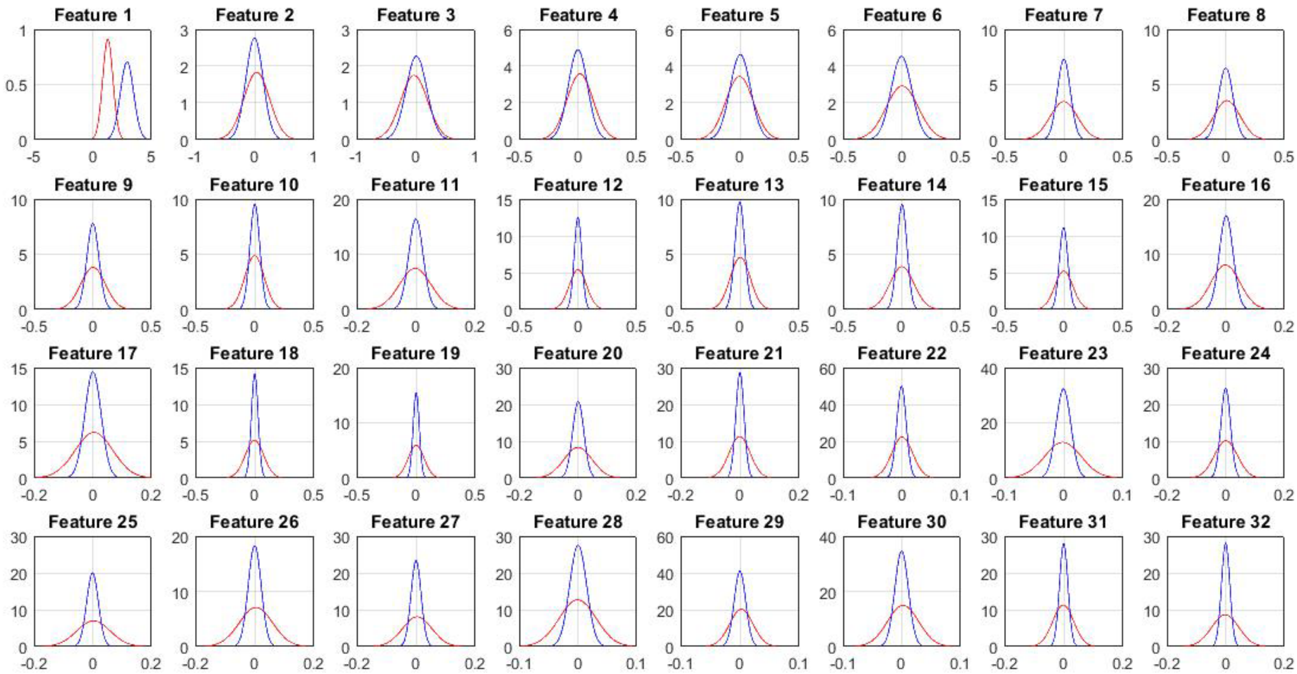


▲Figure 1

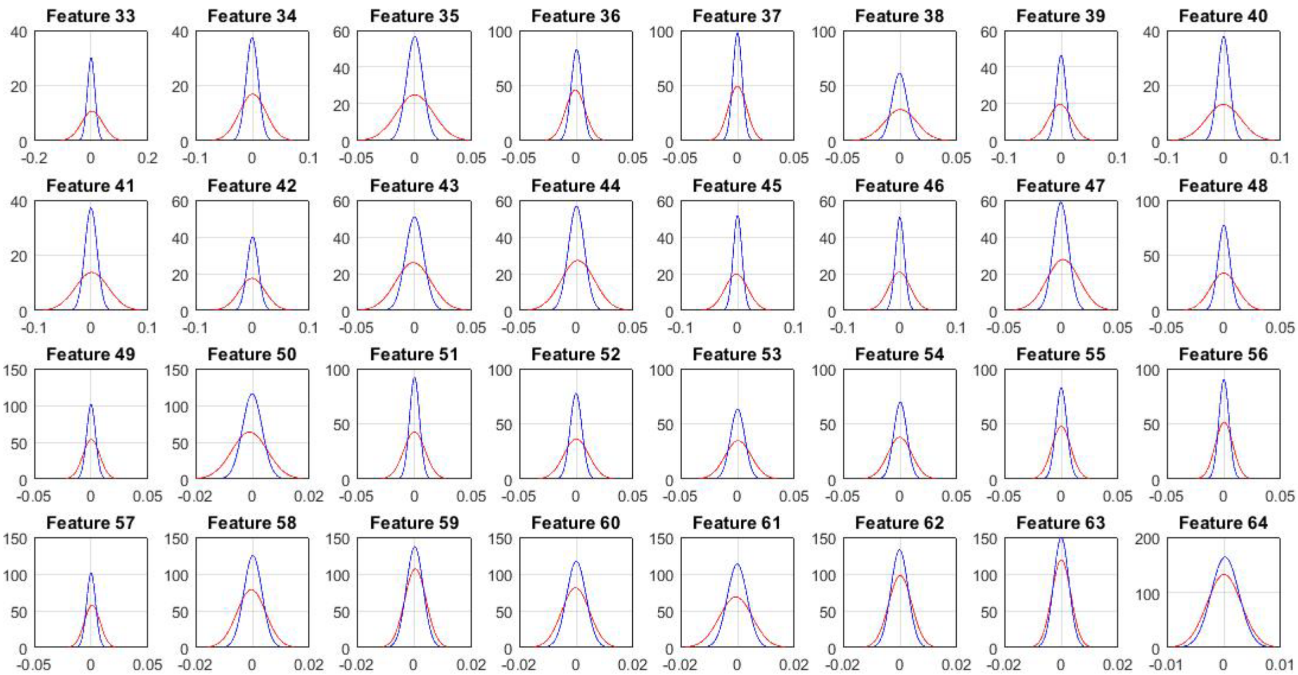
The above calculation is exactly as same as what I did in the homework 1. In homework 1, I estimated the prior probabilities by calculating the relative frequencies of these two classes.

(b)

Assume the ML estimate of the conditional class probabilities $P_{X|Y}(x|cheetach)$ and $P_{X|Y}(x|grass)$ fits the Gaussian distribution. Here, I plot each conditional class probabilities with read line for *class cheetah* and blue line for *class grass* in Figure 2 and Figure 3.

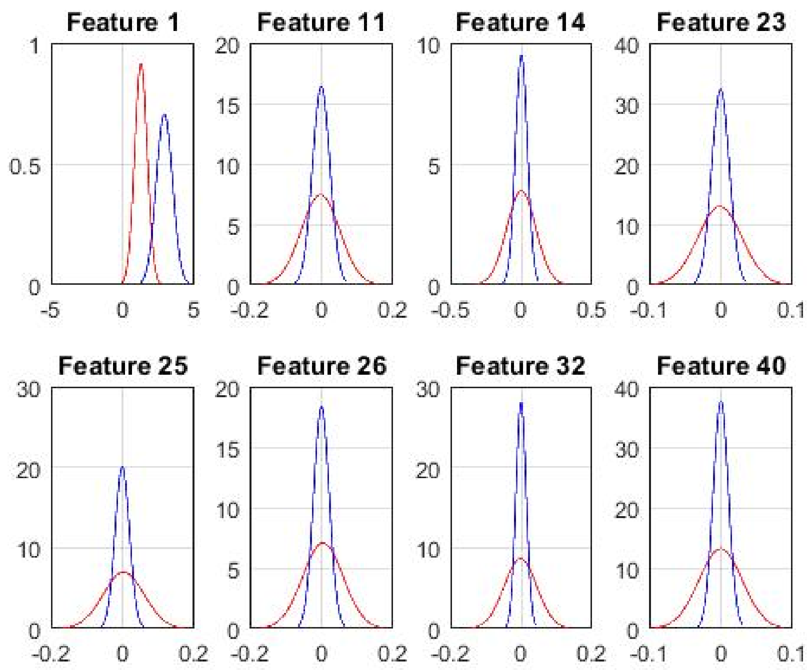


▲Figure 2

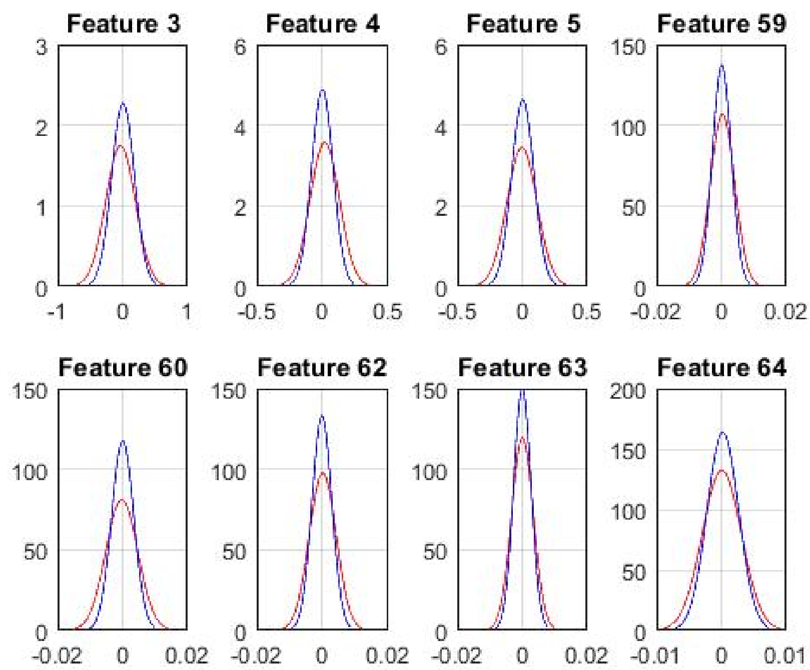


▲Figure 3

By visual inspection, I roughly pick the first 8 best features and the first 8 worst features by comparing the level of distribution spread of each feature because it is not easy to decide the best parts of features using their distribution means. I select [1,11,14,23,25,26,32,40] for my best 8 features. On the other hand, I select [3,4,5,59,60,62,63,64] as my 8 worst features. The marginal densities of the first 8 best features and the worst features are shown in Figure 4 and Figure 5.



▲Figure 4 The best 8 features



▲Figure 5 The worst 8 features

(c)

According to the equations taught in class, multivariate Gaussian is expressed as :

$$P_{X|Y}(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} exp\left\{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right\}$$

For our cases, dimension is either 64 or 8. Then, when there are only two classes, it's convenient to use the sigmoid.

$$i^*(x) = arg\ \max_i g_i(x), \text{ with } g_i(x) = P_{Y|X}(i|x)$$

For Gaussian classes, the posterior probabilities are

$$g_0(x) = \frac{1}{1 + exp\{d_0(x-\mu_0) - d_1(x-\mu_1) + \alpha_0 - \alpha_1\}}$$

$$d_i(x,y) = (x-y)^T \Sigma_i^{-1}(x-y)$$

$$\alpha_1 = \log(2\pi)^d |\Sigma_i| - 2logP_Y(i)$$

And decide the pixel to be "cheetah" if $g_{cheetah}(x) > 0.5$.

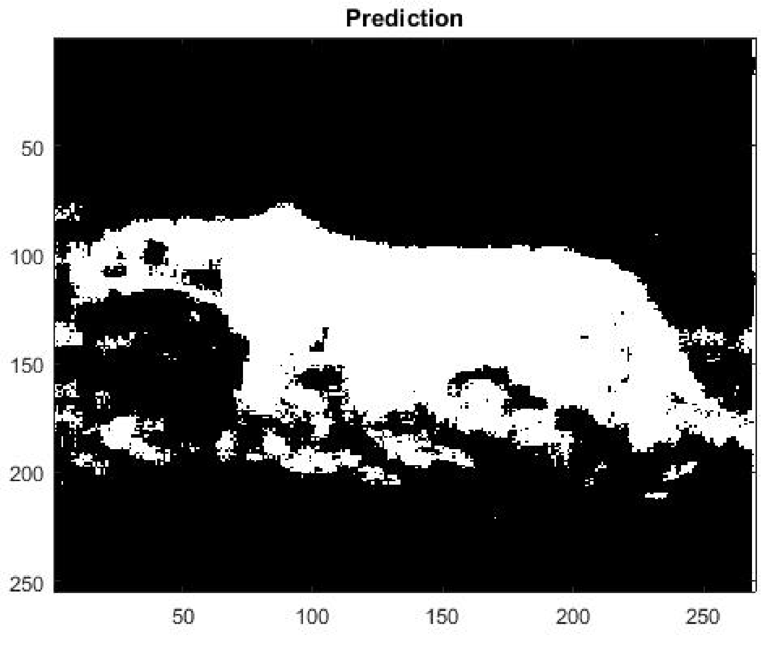
Besides, in my implementation, I adopt the right and top padding with symmetric pattern and make the predicted pixel to be the down right pixel of the 8*8 block.

For the error rate calculation, I would use the following equation to estimate.

$$P(error) = P_{X|Y}(grass|cheetah) \times P_Y(cheetah) + P_{X|Y}(cheetah|grass) \times P_Y(grass)$$

(i) Use the 64-dimensional Gaussians

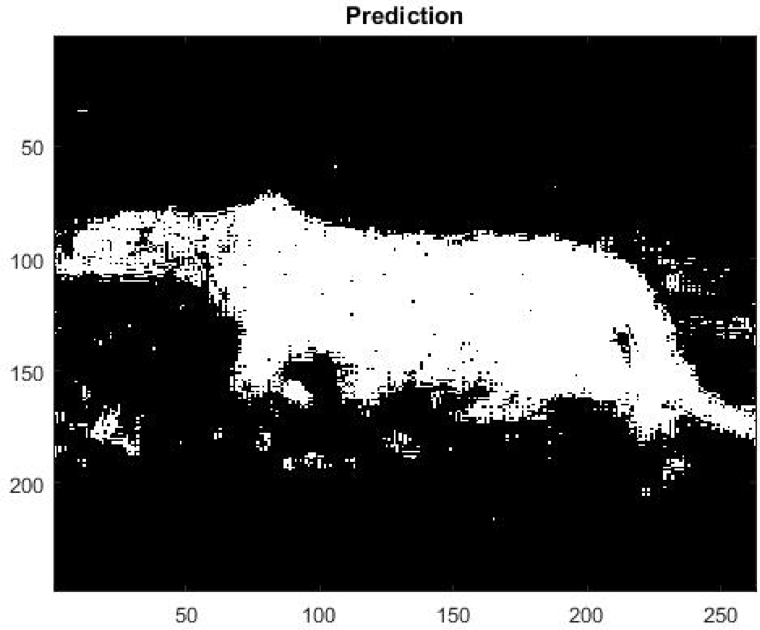
$$P(error) = 0.0753 \times 0.1919 + 0.0978 \times 0.8081 = 0.0935$$



▲Figure 6: Classification mask using 64 features

(ii) Use the 8-dimensional Gaussians associated with the best 8 features

$$P(error) = 0.1018 \times 0.1919 + 0.0388 \times 0.8081 = 0.0509$$



▲Figure 7: Classification mask using the best 8 features

Obviously, from the Figure 6 and Figure 7, it can conclude that using more features does not necessarily perform better prediction. In our case, picking the best 8 features instead creates more accurate classification mask.

MATLAB CODE (For 64-dimension features)

```
close all
clear all
clc

load('TrainingSamplesDCT_8_new.mat')

%% Problem a
[row_f,col_f] = size(TrainsampleDCT_FG);
FG_index=zeros(1,row_f);
[row_b,col_b] = size(TrainsampleDCT_BG);
```

```
BG_index=zeros(1,row_b);

Prior_FG=row_f/(row_f+row_b);        % Calculate Prior
Prior_BG=row_b/(row_f+row_b);

figure;
str = {'cheetah','grass'};
bar([Prior_FG,Prior_BG])
title('Histogram of Prior Probabilities')
xlabel('Class')
ylabel('Prior')
set(gca, 'XTickLabel',str),


disp('Prior Probability (cheetah):')
disp(Prior_FG)
disp('Prior Probability (grass):')
disp(Prior_BG)

%% Problem b
% calculate for each column's mean
mu_fg=sum(TrainsampleDCT_FG)/row_f;
mu_bg=sum(TrainsampleDCT_BG)/row_b;
% calculate for each column's std
std_fg=std(TrainsampleDCT_FG);
std_bg=std(TrainsampleDCT_BG);


for i=1:col_b
   % create Gaussian function for each column
   % I use overall 6 std to be my x-axis
   margin_fg_axis(:,i)=[(mu_fg(i)-
3*std_fg(i)):(std_fg(i)/50):(mu_fg(i)+3*std_fg(i))];
   margin_FG(:,i)=normpdf(margin_fg_axis(:,i),mu_fg(i),std_fg(i));

   margin_bg_axis(:,i)=[(mu_bg(i)-
3*std_bg(i)):(std_bg(i)/50):(mu_bg(i)+3*std_bg(i))];
   margin_BG(:,i)=normpdf(margin_bg_axis(:,i),mu_bg(i),std_bg(i));

end
```

```
% plot the marginal density into 8*8 plots with the dimension from 1-64
for i=1:2
    figure;
    for j=1:32
        subplot(4,8,j)
        plot(margin_fg_axis(:,(i-1)*32+j),margin_FG(:,(i-1)*32+j),'-
r',margin_bg_axis(:,(i-1)*32+j),margin_BG(:,(i-1)*32+j),'-b')
        grid on
        title(['Feature ',num2str((i-1)*32+j)])
    end
end
% plot the best 8 feature marginal densities
best = [1,11,14,23,25,26,32,40];
figure;
for i=1:8
    subplot(2,4,i)
    plot(margin_fg_axis(:,best(i)),margin_FG(:,best(i)),'-
r',margin_bg_axis(:,best(i)),margin_BG(:,best(i)),'-b')
    grid on
    title(['Feature ',num2str(best(i))])

end
% plot the worst 8 feature marginal densities
worst=[3,4,5,59,60,62,63,64];
figure;
for i=1:8
    subplot(2,4,i)
    plot(margin_fg_axis(:,worst(i)),margin_FG(:,worst(i)),'-
r',margin_bg_axis(:,worst(i)),margin_BG(:,worst(i)),'-b')
    grid on
    title(['Feature ',num2str(worst(i))])

end
%% Problem c
% Load Test sample
Img_ori=imread('cheetah.bmp');          %Use zero padding for edge & corner
Img = padarray(Img_ori,[7 7],'symmetric','pre');     %classified pixel:right
bottom
```

```
Img=im2double(Img);
[row,col]=size(Img);

% build 64-dimension Gaussian
covar_fg=cov(TrainsampleDCT_FG);
covar_bg=cov(TrainsampleDCT_BG);

% Load Zig-Zag pattern
Zigzag=load('Zig-Zag Pattern.txt');
Zigzag=Zigzag+1;

A=zeros((row-7)*(col-7),64);
index=1;
for i=1:row-7
    for j=1:col-7
        Field=Img(i:i+7,j:j+7);
        DCT=dct2(Field);
        DCT_64(Zigzag)=DCT;  % turn 8*8 into 1*64 with Zigzag pattern
        A(index,:)=DCT_64;
        index=index+1;
    end
end

alp_fg=log(((2*pi)^64)*det(covar_fg))-2*log(Prior_FG);
alp_bg=log(((2*pi)^64)*det(covar_bg))-2*log(Prior_BG);
g_fg=zeros((col-7)*(row-7),1);
g_bg=zeros((col-7)*(row-7),1);
temp_dxy_fg=zeros((col-7)*(row-7),1);
temp_dxy_bg=zeros((col-7)*(row-7),1);

predict=zeros(1,(col-7)*(row-7));
for index=1:(col-7)*(row-7)
    temp_dxy_fg(index)=(A(index,:)-mu_fg) * (inv(covar_fg)* (A(index,:)-
mu_fg)');
    temp_dxy_bg(index)=(A(index,:)-mu_bg) * (inv(covar_bg)* (A(index,:)-
mu_bg)');
    g_fg(index)=1 / (1+ exp( temp_dxy_fg(index) - temp_dxy_bg(index) + alp_fg -
alp_bg));
    g_bg(index)=1 / (1+ exp( temp_dxy_bg(index) - temp_dxy_fg(index) + alp_bg -
alp_fg));
```

```
    if g_fg(index)>0.5
        predict(1,index)=1;
    end
    index=index+1;
end

predict_2d=reshape(predict,[col-7,row-7]);
predict_2d=predict_2d';

figure;
imagesc(predict_2d);
title('Prediction')
colormap(gray(255));

%% Calculate the Probability of Error
% Load Ground Truth
Gt=imread('cheetah_mask.bmp');
Gt=im2double(Gt);

Gt_FG=0;
Gt_BG=0;
for i=1:row-7
    for j=1:col-7
        if Gt(i,j)==1
            Gt_FG=Gt_FG+1;
        end
        if Gt(i,j)==0
            Gt_BG=Gt_BG+1;
        end
    end
end

Errors_FG=0;
Errors_BG=0;
for i=1:row-7
    for j=1:col-7
        if Gt(i,j)==1 && predict_2d(i,j)==0 % FG pixels, misclassifcied as BG
            Errors_FG=Errors_FG+1;
        end
```

```
        if Gt(i,j)==0 && predict_2d(i,j)==1
            Errors_BG=Errors_BG+1;
        end
    end
end

Errors_FG_p=Errors_FG/Gt_FG;    %Type II (False Negative)
Errors_BG_p=Errors_BG/Gt_BG;    %Type I  (False Positive)

Errors=Errors_FG_p*Prior_FG + Errors_BG_p*Prior_BG;

disp('Error Rate (%):')
disp(Errors*100)
```

MATLAB CODE (For 8-dimension features)

```
close all
clear all
clc

load('TrainingSamplesDCT_8_new.mat')

[row_f,col_f] = size(TrainsampleDCT_FG);
FG_index=zeros(1,row_f);
[row_b,col_b] = size(TrainsampleDCT_BG);
BG_index=zeros(1,row_b);

Prior_FG=row_f/(row_f+row_b);        % Calculate Prior
Prior_BG=row_b/(row_f+row_b);



best = [1,11,14,23,25,26,32,40];

best_BG=TrainsampleDCT_BG(:,best);
best_FG=TrainsampleDCT_FG(:,best);

mu_fg=sum(best_FG)/row_f;  % calculate for each column's mean
mu_bg=sum(best_BG)/row_b;
```

```
% Load Test sample
Img_ori=imread('cheetah.bmp');          %Use zero padding for edge & corner
Img = padarray(Img_ori,[7 7],'symmetric','pre');    %classified pixel:right
bottom
%Img = padarray(Img_ori,[7 7],'post');    %classified pixel:left top
Img=im2double(Img_ori);
[row,col]=size(Img);

% build 8-dimension Gaussian
covar_fg=cov(best_FG);
covar_bg=cov(best_BG);

% Load Zig-Zag pattern
Zigzag=load('Zig-Zag Pattern.txt');
Zigzag=Zigzag+1;

A=zeros((row-7)*(col-7),8);
index=1;
for i=1:row-7
    for j=1:col-7
        Field=Img(i:i+7,j:j+7);
        DCT=dct2(Field);
        DCT_64(Zigzag)=DCT;  % turn 8*8 into 1*64 with Zigzag pattern
        A(index,:)=DCT_64(best);
        index=index+1;
    end
end

alp_fg=log(((2*pi)^8)*det(covar_fg))-2*log(Prior_FG);
alp_bg=log(((2*pi)^8)*det(covar_bg))-2*log(Prior_BG);
g_fg=zeros((col-7)*(row-7),1);
g_bg=zeros((col-7)*(row-7),1);
temp_dxy_fg=zeros((col-7)*(row-7),1);
temp_dxy_bg=zeros((col-7)*(row-7),1);

predict=zeros(1,(col-7)*(row-7));
for index=1:(col-7)*(row-7)
    temp_dxy_fg(index)=(A(index,:)-mu_fg) * (inv(covar_fg)* ((A(index,:)-
mu_fg)'));
```

```
    temp_dxy_bg(index)=(A(index,:)-mu_bg) * (inv(covar_bg)* ((A(index,:)-
mu_bg)'));
    g_fg(index)=1 / (1+ exp( temp_dxy_fg(index) - temp_dxy_bg(index) + alp_fg -
alp_bg));
    g_bg(index)=1 / (1+ exp( temp_dxy_bg(index) - temp_dxy_fg(index) + alp_bg -
alp_fg));

    if g_fg(index)>0.5
        predict(1,index)=1;
    end
    index=index+1;
end

predict_2d=reshape(predict,[col-7,row-7]);
predict_2d=predict_2d';

figure;
imagesc(predict_2d);
title('Prediction')
colormap(gray(255));

% Calculate the Probability of Error
% Load Ground Truth
Gt=imread('cheetah_mask.bmp');
Gt=im2double(Gt);

Gt_FG=0;
Gt_BG=0;
for i=1:row
    for j=1:col
        if Gt(i,j)==1
            Gt_FG=Gt_FG+1;
        end
        if Gt(i,j)==0
            Gt_BG=Gt_BG+1;
        end
    end
end

Errors_FG=0;
```

```
Errors_BG=0;
for i=1:row-7
    for j=1:col-7
        if Gt(i,j)==1 && predict_2d(i,j)==0 % FG pixels, misclassifcied as BG
            Errors_FG=Errors_FG+1;
        end
        if Gt(i,j)==0 && predict_2d(i,j)==1
            Errors_BG=Errors_BG+1;
        end
    end
end

Errors_FG_p=Errors_FG/Gt_FG;    %Type II (False Negative)
Errors_BG_p=Errors_BG/Gt_BG;    %Type I (False Positive)

Errors=Errors_FG_p*Prior_FG + Errors_BG_p*Prior_BG;

disp('Error Rate (%):')
disp(Errors*100)
```