

Homework Set Two

ECE 271A

Name : Hsiu-Wen Yen

ID : A59010599

a) From problem 2, we get $\pi_j = \frac{c_j}{n}$.

Therefore, $\text{Prior_Probability}(\text{cheetah}) = \text{Counts}(\text{cheetah}) / \text{Total Counts}$.

The result is same as last week.

What I did last week is intuitive for using frequency of cheetah as probability of cheetah, which is same with result of problem 2.

Ans:

$P(\text{cheetah}) = 0.1919$

$P(\text{Grass}) = 0.8081$

```
1 clear; clc;
2 % load trainging sample and image
3 load('TrainingSamplesDCT_8_new.mat');
4 cheetah = imread('cheetah.bmp');
5 % calculate prior probabilities of cheetah and grass
6 pixel_total_count = size(TrainsampleDCT_FG, 1) + size(TrainsampleDCT_BG, 1);
7 prior_Pcheetah = size(TrainsampleDCT_FG, 1) / pixel_total_count;
8 prior_Pgrass = size(TrainsampleDCT_BG, 1) / pixel_total_count;
9
```

Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> prior_Pcheetah
```

```
prior_Pcheetah =
```

```
0.1919
```

```
>> prior_Pgrass
```

```
prior_Pgrass =
```

```
0.8081
```

b) First calculate sigma and mean for each dct coefficient.

```

11 % estimate marginal densities
12 % front ground
13 mean_FG = mean(TrainsampleDCT_FG);
14 variance_FG = var(TrainsampleDCT_FG);
15 sigma_FG = sqrt(variance_FG);
16 sum_FG = sum(TrainsampleDCT_FG);
17 % back ground
18 mean_BG = mean(TrainsampleDCT_BG);
19 variance_BG = var(TrainsampleDCT_BG);
20 sigma_BG = sqrt(variance_BG);
21 sum_BG = sum(TrainsampleDCT_BG);

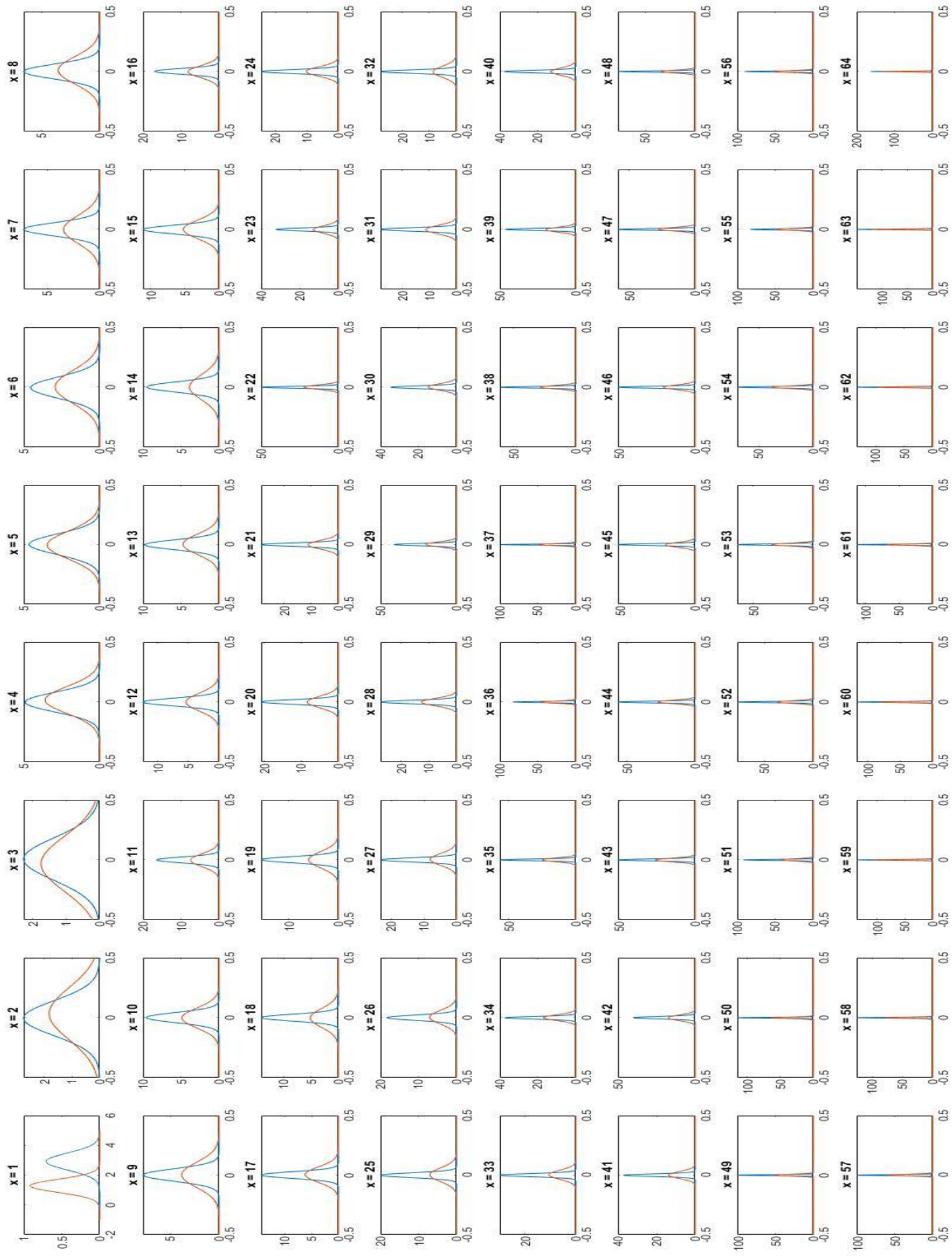
```

Plot 64 for marginal densities for the two classes.

```

24 for i = 1 : 64
25     % check negative definite
26     % calculate Hessian matrix [A B; B C] for Front ground
27     a_FG = - size(TrainsampleDCT_FG,1) / (sigma_FG(i))^2;
28     b_FG = -2 * (sum_FG(i) - size(TrainsampleDCT_FG,1) * mean_FG(i)) / (sigma_FG(i))^3;
29     c_FG = size(TrainsampleDCT_FG,1) / (sigma_FG(i))^2 - 3 / (sigma_FG(i))^3 * size(TrainsampleDCT_FG,1);
30     % calculate Hessian matrix [A B; B C] for Back ground
31     a_BG = - size(TrainsampleDCT_BG,1) / (sigma_BG(i))^2;
32     b_BG = -2 * (sum_BG(i) - size(TrainsampleDCT_BG,1) * mean_BG(i)) / (sigma_BG(i))^3;
33     c_BG = size(TrainsampleDCT_BG,1) / (sigma_BG(i))^2 - 3 / (sigma_BG(i))^3 * size(TrainsampleDCT_BG,1);
34     % using eigenvalue of Hessian matrix to check negative definite
35     H = [a_FG, b_FG; b_FG, c_FG];
36     e = eig(H);
37     if all(e > 0) % if all eigenvalue is negative, Hessian matrix is negative definite
38         continue; % skip plot when one of eigenvalue is positive
39     end
40     H = [a_BG, b_BG; b_BG, c_BG];
41     e = eig(H);
42     if all(e > 0) % if all eigenvalue is negative, Hessian matrix is negative definite
43         continue; % skip plot when one of eigenvalue is positive
44     end
45
46     % plot BG
47     % **NOTE** for first dct coefficient, mean is bigger than other.
48     % Therefore, it has different x interval.
49     if (i == 1)
50         x = -1:0.001:5;
51     else
52         x = -0.5:0.001:0.5;
53     end
54     y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) / sigma_BG(i)).^2);
55     subplot(8, 8, i);
56     txt = "x = " + int2str(i);
57     plot(x,y);
58     hold on;
59     % plot FG
60     y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) / sigma_FG(i)).^2);
61     plot(x,y);
62     title(txt);
63 end

```



From picture above, I think $x = [1 \ 6 \ 7 \ 8 \ 9 \ 10 \ 12 \ 13]$ have clear clarification, you can see two curves distinctly. For $x = [2 \ 3 \ 4 \ 5]$, two curves almost overlap each other.

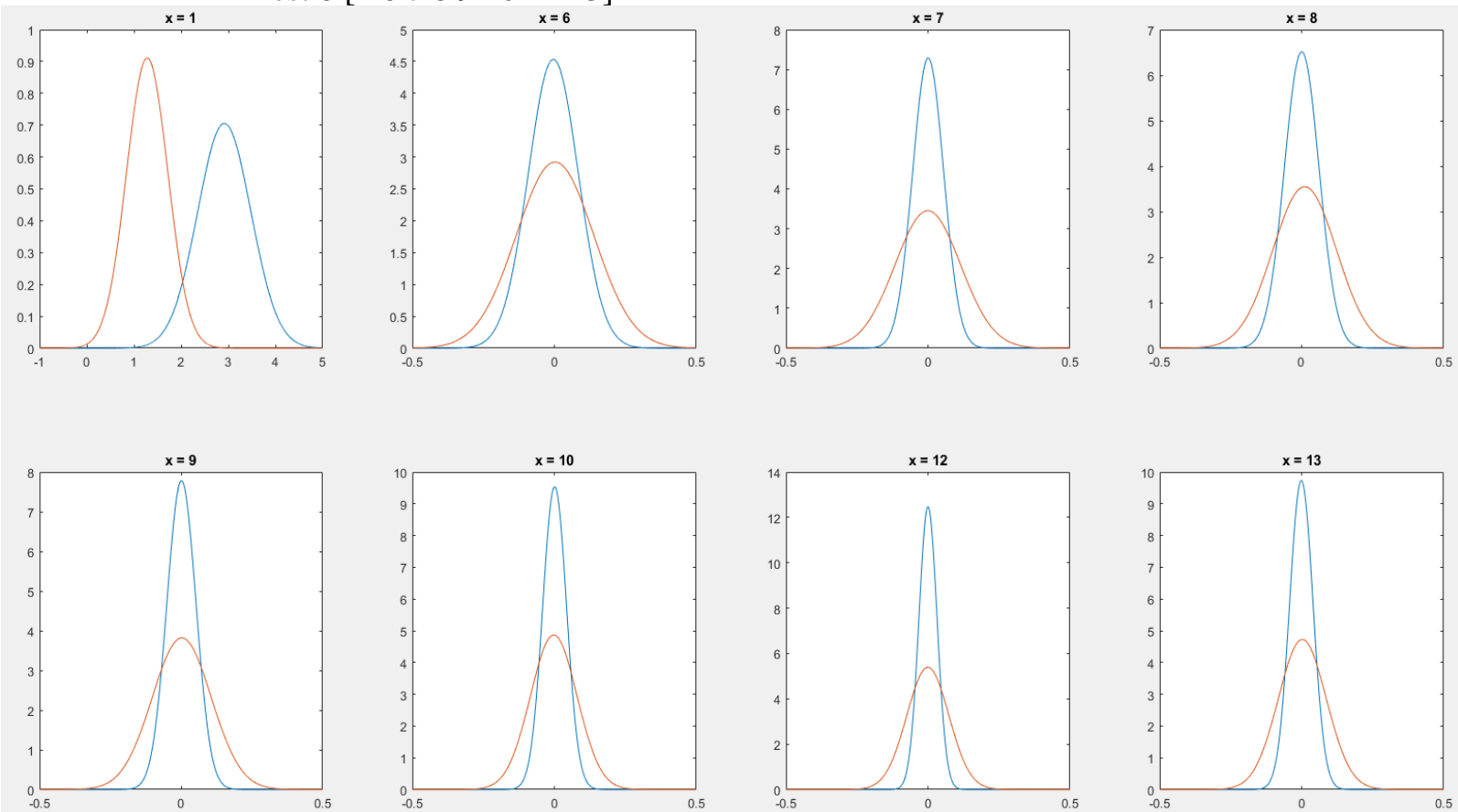
On the other hand, $x = [57 \ 58 \ 59 \ 60 \ 61 \ 62 \ 63 \ 64]$ have worst clarification, the range for identification is so small.

```

65 %% plot best and worst section
66 best_idx = [1 6 7 8 9 10 12 13];
67 worst_idx = [57, 58, 59, 60, 61, 62, 63, 64];
68 % plot best
69 figure(2);
70 for counter = 1 : size(best_idx, 2)
71     i = best_idx(counter);
72     % plot BG
73     if (i == 1)
74         x = -1:0.001:5;
75     else
76         x = -0.5:0.001:0.5;
77     end
78     y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) / sigma_BG(i)).^2);
79     subplot(2, 4, counter);
80     txt = "x = " + int2str(i);
81     plot(x,y);
82     hold on;
83     % plot FG
84     y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) / sigma_FG(i)).^2);
85     plot(x,y);
86     title(txt);
87 end

```

Best 8 [1 6 7 8 9 10 12 13]

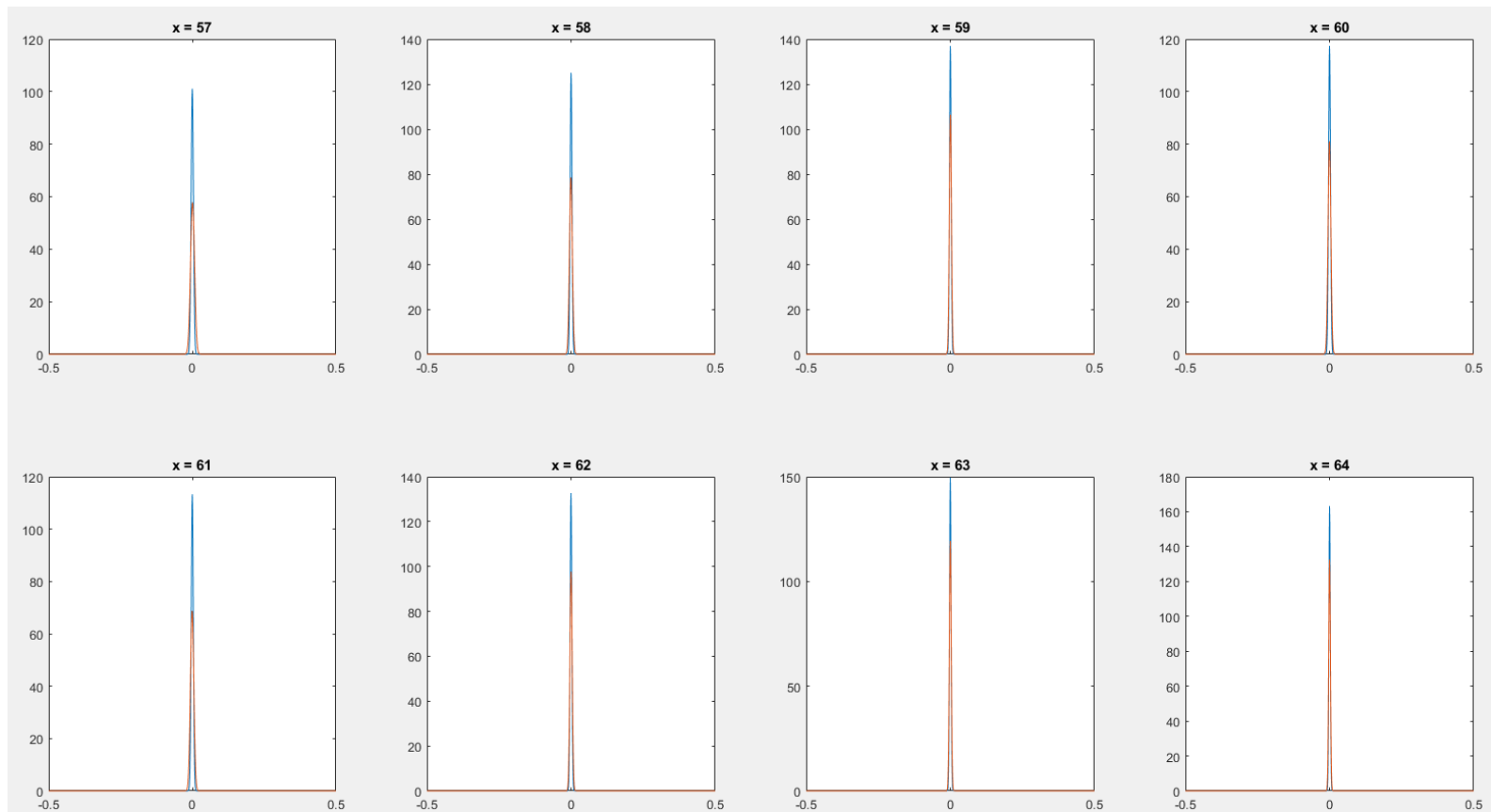


```

88 % plot worst|
89 figure(3);
90 for counter = 1 : size(worst_idx, 2)
91     i = worst_idx(counter);
92     % plot BG
93     if (i == 1)
94         x = -1:0.001:5;
95     else
96         x = -0.5:0.001:0.5;
97     end
98     y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) / sigma_BG(i)).^2);
99     subplot(2, 4, counter);
100    txt = "x = " + int2str(i);
101    plot(x,y);
102    hold on;
103    % plot FG
104    y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) / sigma_FG(i)).^2);
105    plot(x,y);
106    title(txt);
107 end

```

Worst 8 [57 58 59 60 61 62 63 64]



c) First calculate 64D covariance

```
109     %% 64D feature section
110     % calculate covariance for 64D for FG
111     covariance_64_FG = cov(TrainsampleDCT_FG);
112     covariance_64_BG = cov(TrainsampleDCT_BG);
113
114     % create output mask image array
115     row_size = size(cheetah, 1);
116     column_size = size(cheetah, 2);
117     A_64 = zeros(row_size, column_size);
118
```

According BDR to create cheetah mask for 64D.

```
119 % using 8 * 8 blocks to represent the left top pixel
120 for rows = 1 : row_size - 8 + 1
121     for columns = 1 : column_size - 8 + 1
122         block = cheetah(rows:rows+7, columns:columns+7);
123         block = dct2(block);
124         x = expand_zigzag(block);
125         % for FG and BG
126         p_FG = (-0.5*(x - mean_FG)/ covariance_64_FG * (x - mean_FG).') - log(sqrt(det(covariance_64_FG)*(2*pi)^64)) + log(prior_Pcheetah);
127
128         p_BG = (-0.5*(x - mean_BG)/ covariance_64_BG * (x - mean_BG).') - log(sqrt(det(covariance_64_BG)*(2*pi)^64)) + log(prior_Pgrass);
129
130         if (p_BG > p_FG)
131             A_64(rows, columns) = 0;
132         else
133             A_64(rows, columns) = 1;
134         end
135     end
136 end
137 figure(4);
138 imagesc(A_64);
139 colormap(gray(255));
```

For best 8d feature, do the same thing.

Calculate 8d covariance and mean.

```
140 %% 8D feature section
141 % extrac required feature from training set
142 for j = 1 : 8
143     required_8d_FG(:,j) = TrainsampleDCT_FG(:, best_idx(j));
144     required_8d_BG(:,j) = TrainsampleDCT_BG(:, best_idx(j));
145 end
146
147 % calculate covariance and mean for 8D
148 covariance_8_FG = cov(required_8d_FG);
149 covariance_8_BG = cov(required_8d_BG);
150 mean_8d_FG = mean(required_8d_FG);
151 mean_8d_BG = mean(required_8d_BG);
```

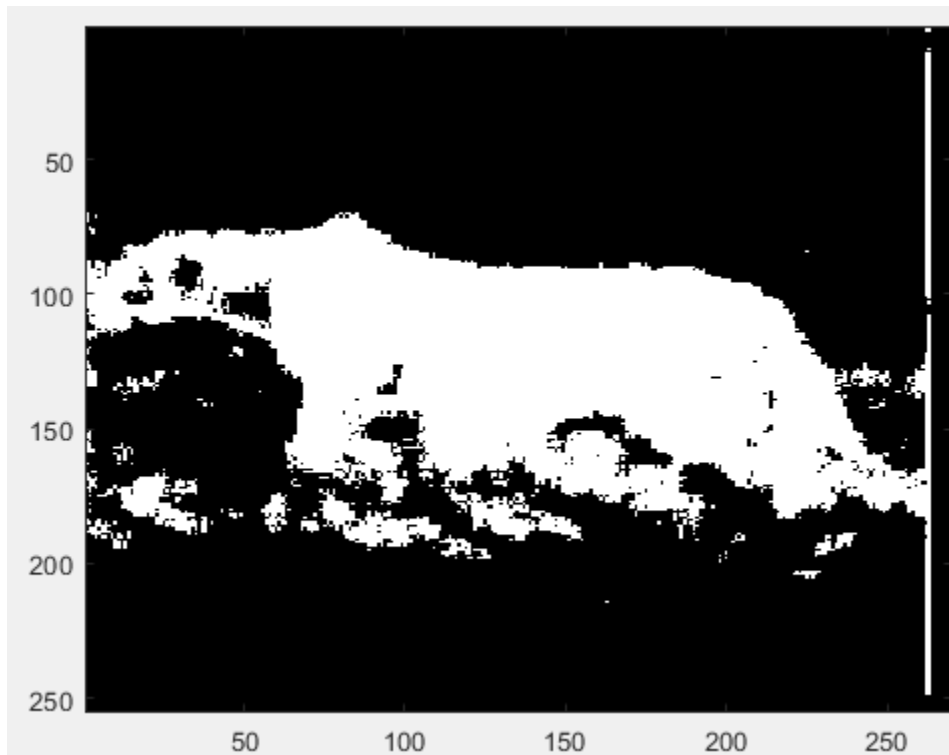
According BDR to create cheetah mask for 8D.

```
153 A_8 = zeros(row_size, column_size);
154 % using 8 * 8 blocks to represent the left top pixel
155 for rows = 1 : row_size - 8 + 1
156     for columns = 1 : column_size - 8 + 1
157         block = cheetah(rows:rows+7, columns:columns+7);
158         block = dct2(block);
159         x = expand_zigzag(block);
160         % for FG and BG
161         % get feature value
162         for j = 1 : 8
163             x_8d(j) = x(best_idx(j));
164         end
165         p_FG = (-0.5*(x_8d - mean_8d_FG)/ covariance_8_FG * (x_8d - mean_8d_FG).') - log(sqrt(det(covariance_8_FG)*(2*pi)^64)) + log(prior_Pcheetah);
166         p_BG = (-0.5*(x_8d - mean_8d_BG)/ covariance_8_BG * (x_8d - mean_8d_BG).') - log(sqrt(det(covariance_8_BG)*(2*pi)^64)) + log(prior_Pgrass);
167         if (p_BG > p_FG)
168             A_8(rows, columns) = 0;
169         else
170             A_8(rows, columns) = 1;
171         end
172     end
173 end
174 figure(5);
175 imagesc(A_8);
176 colormap(gray(255));
```

For estimating probability of error, we need to count how many pixels aren't correct, and then divide it with total pixel counts.

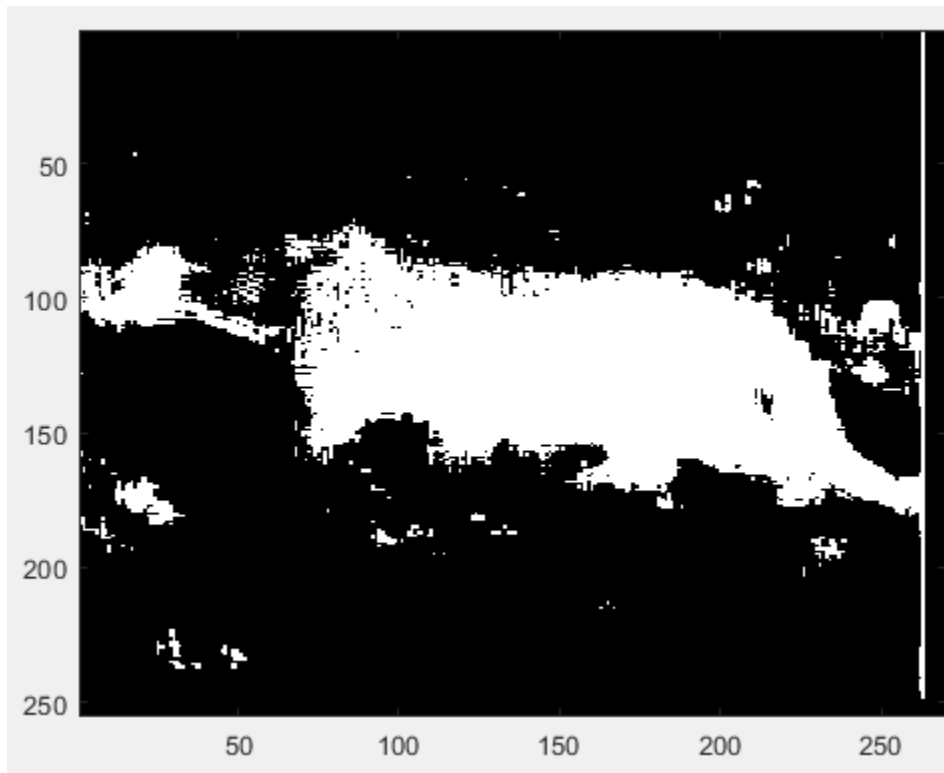
```
178 %% error
179 % load cheetah mask.bmp
180 truth = imread("cheetah_mask.bmp");
181 % calculate last meaningful index of row and column
182 last_row = size(cheetah, 1) - 8 + 1;
183 last_column = size(cheetah, 2) - 8 + 1;
184 % error for 64d
185 truth = double(truth(1 : last_row, 1 : last_column) / 255);
186 A_64 = A_64(1 : last_row, 1 : last_column);
187 err = truth - A_64;
188 err = abs(err);
189 probability_error_64d = sum(err, 'all') / (last_row*last_column);
190 % error for 8d
191 A_8 = A_8(1 : last_row, 1 : last_column);
192 err = truth - A_8;
193 err = abs(err);
194 probability_error_8d = sum(err, 'all') / (last_row*last_column);|
```

64d Result:



Error : 0.094060468539188

8d Result:



Error : 0.063013614620385

```
Command Window
New to MATLAB? See resources for Getting Started

>> probability_error_64d

probability_error_64d =

    0.0941

>> probability_error_8d

probability_error_8d =

    0.0630

fx >> |
```

Result of 64d is worse than 8d because 64d contains too many meaningless features that influence the decision rule. On the other hand, 8d only took useful features for calculating probability, therefore, result is much better than 8d.

I think it is kind of overfitting situation for 64d.

Full code review:

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

% estimate marginal densities
% front ground
mean_FG = mean(TrainsampleDCT_FG);
variance_FG = var(TrainsampleDCT_FG);
sigma_FG = sqrt(variance_FG);
sum_FG = sum(TrainsampleDCT_FG);
% back ground
mean_BG = mean(TrainsampleDCT_BG);
variance_BG = var(TrainsampleDCT_BG);
sigma_BG = sqrt(variance_BG);
sum_BG = sum(TrainsampleDCT_BG);

figure(1);
for i = 1 : 64
    % check negative definite
    % calculate Hessian matrix [A B; B C] for Front ground
    a_FG = - size(TrainsampleDCT_FG,1) / (sigma_FG(i))^2;
    b_FG = -2 * (sum_FG(i) - size(TrainsampleDCT_FG,1) * mean_FG(i)) /
(sigma_FG(i))^3;
    c_FG = size(TrainsampleDCT_FG,1) / (sigma_FG(i))^2 - 3 / (sigma_FG(i))^3 *
size(TrainsampleDCT_FG,1);
    % calculate Hessian matrix [A B; B C] for Back ground
    a_BG = - size(TrainsampleDCT_BG,1) / (sigma_BG(i))^2;
    b_BG = -2 * (sum_BG(i) - size(TrainsampleDCT_BG,1) * mean_BG(i)) /
(sigma_BG(i))^3;
    c_BG = size(TrainsampleDCT_BG,1) / (sigma_BG(i))^2 - 3 / (sigma_BG(i))^3 *
size(TrainsampleDCT_BG,1);
    % using eigenvalue of Hessisan matrix to check negative definite
    H = [a_FG, b_FG; b_FG, c_FG];
    e = eig(H);
    if all(e > 0) % if all eigenvalue is negative, Hessian matrix is negative
definite
        continue; % skip plot when one of eigenvalue is positive
    end
    H = [a_BG, b_BG; b_BG, c_BG];
    e = eig(H);
    if all(e > 0) % if all eigenvalue is negative, Hessian matrix is negative
definite
        continue; % skip plot when one of eigenvalue is positive
    end

    % plot BG
    % **NOTE** for first dct coefficient, mean is bigger than other.
```

```

    % Therefore, it has different x interval.
    if (i == 1)
        x = -1:0.001:5;
    else
        x = -0.5:0.001:0.5;
    end
    y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) /
sigma_BG(i)).^2);
    subplot(8, 8, i);
    txt = "x = " + int2str(i);
    plot(x,y);
    hold on;
    % plot FG
    y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) /
sigma_FG(i)).^2);
    plot(x,y);
    title(txt);
end

%% plot best and worst section
best_idx = [1 6 7 8 9 10 12 13];
worst_idx = [57, 58, 59, 60, 61, 62, 63, 64];
% plot best
figure(2);
for counter = 1 : size(best_idx, 2)
    i = best_idx(counter);
    % plot BG
    if (i == 1)
        x = -1:0.001:5;
    else
        x = -0.5:0.001:0.5;
    end
    y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) /
sigma_BG(i)).^2);
    subplot(2, 4, counter);
    txt = "x = " + int2str(i);
    plot(x,y);
    hold on;
    % plot FG
    y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) /
sigma_FG(i)).^2);
    plot(x,y);
    title(txt);
end
% plot worst
figure(3);
for counter = 1 : size(worst_idx, 2)
    i = worst_idx(counter);
    % plot BG
    if (i == 1)
        x = -1:0.001:5;
    else
        x = -0.5:0.001:0.5;
    end
end

```

```

        y = 1 / (sigma_BG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_BG(i)) /
sigma_BG(i)).^2);
        subplot(2, 4, counter);
        txt = "x = " + int2str(i);
        plot(x,y);
        hold on;
        % plot FG
        y = 1 / (sigma_FG(i) * sqrt(2 * pi)) * exp(- 0.5 * ((x - mean_FG(i)) /
sigma_FG(i)).^2);
        plot(x,y);
        title(txt);
    end

%% 64D feature section
% calculate covariance for 64D for FG
covariance_64_FG = cov(TrainsampleDCT_FG);
covariance_64_BG = cov(TrainsampleDCT_BG);

% 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 = (-0.5*(x - mean_FG)/ covariance_64_FG * (x - mean_FG).') -
log(sqrt(det(covariance_64_FG)*(2*pi)^64)) + log(prior_Pcheetah);

        p_BG = (-0.5*(x - mean_BG)/ covariance_64_BG * (x - mean_BG).') -
log(sqrt(det(covariance_64_BG)*(2*pi)^64)) + log(prior_Pgrass);

        if (p_BG > p_FG)
            A_64(rows, columns) = 0;
        else
            A_64(rows, columns) = 1;
        end
    end
end
figure(4);
imagesc(A_64);
colormap(gray(255));
%% 8D feature section
% extrac required feature from training set
for j = 1 : 8
    required_8d_FG(:,j) = TrainsampleDCT_FG(:, best_idx(j));
    required_8d_BG(:,j) = TrainsampleDCT_BG(:, best_idx(j));
end

% calculate covariance and mean for 8D
covariance_8_FG = cov(required_8d_FG);

```

```

covariance_8_BG = cov(required_8d_BG);
mean_8d_FG = mean(required_8d_FG);
mean_8d_BG = mean(required_8d_BG);

A_8 = 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
        % get feature value
        for j = 1 : 8
            x_8d(j) = x(best_idx(j));
        end
        p_FG = (-0.5*(x_8d - mean_8d_FG)/ covariance_8_FG * (x_8d - mean_8d_FG).') -
log(sqrt(det(covariance_8_FG)*(2*pi)^64)) + log(prior_Pcheetah);
        p_BG = (-0.5*(x_8d - mean_8d_BG)/ covariance_8_BG * (x_8d - mean_8d_BG).') -
log(sqrt(det(covariance_8_BG)*(2*pi)^64)) + log(prior_Pgrass);
        if (p_BG > p_FG)
            A_8(rows, columns) = 0;
        else
            A_8(rows, columns) = 1;
        end
    end
end
figure(5);
imagesc(A_8);
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);
% error for 8d
A_8 = A_8(1 : last_row, 1 : last_column);
err = truth - A_8;
err = abs(err);
probability_error_8d = sum(err,'all') / (last_row*last_column);

```

```

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
```
