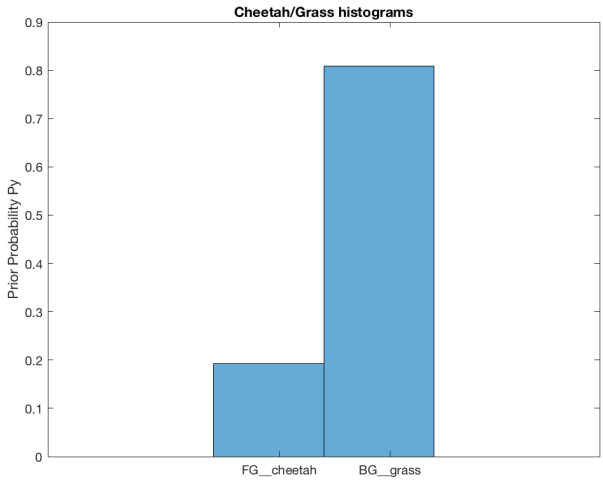
Statistical Learning – Multiple Gaussian Distribution

Yifan Xu
October 30, 2018

Goal: to segment the" cheetah" image into its two components, cheetah(foreground) and grass(background).

a) Using the training data in TrainingSamplesDCT 8.mat compute the histogram estimate of the prior $P_Y(i)$, $i \in \{cheetah, grass\}$. Using the results of problem 2 compute the maximum likelihood estimate for the prior probabilities. Compare the result with the estimates that you obtained last week. If they are the same, interpret what you did last week. If they are different, explain the differences.

Answer:



From the results of problem 2, we know that the Maximum likelihood estimate for the prior probabilities is

$$\pi_j^* = = \frac{c_j}{n}$$

where c_j is the number of the class which has been observed. N is the total number of the training set.

For this question, $c_{cheetah} = 250$, $c_{grass} = 1053$. Therefore, we can get the prior probabilities of cheetah and grass

$$\text{Py(cheetah)} = = \frac{250}{250 + 1053} = 0.1919$$

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

Compared to the estimates that I obtained last week, they are the same. This is because that I did the same thing last week. I divided the quantities of cheetah(grass) by the total number of the training set which led us to get the prior probabilities, just like the maximum likelihood estimate.

b) Create 64 plots with the marginal densities for the two classes $P_{X_k|Y}(X_k|cheetah)$ and $P_{X_k|Y}(X_k|grass)$. What you think are the best 8 features for classification purposes and what you think are the worst 8 features.

Answer:

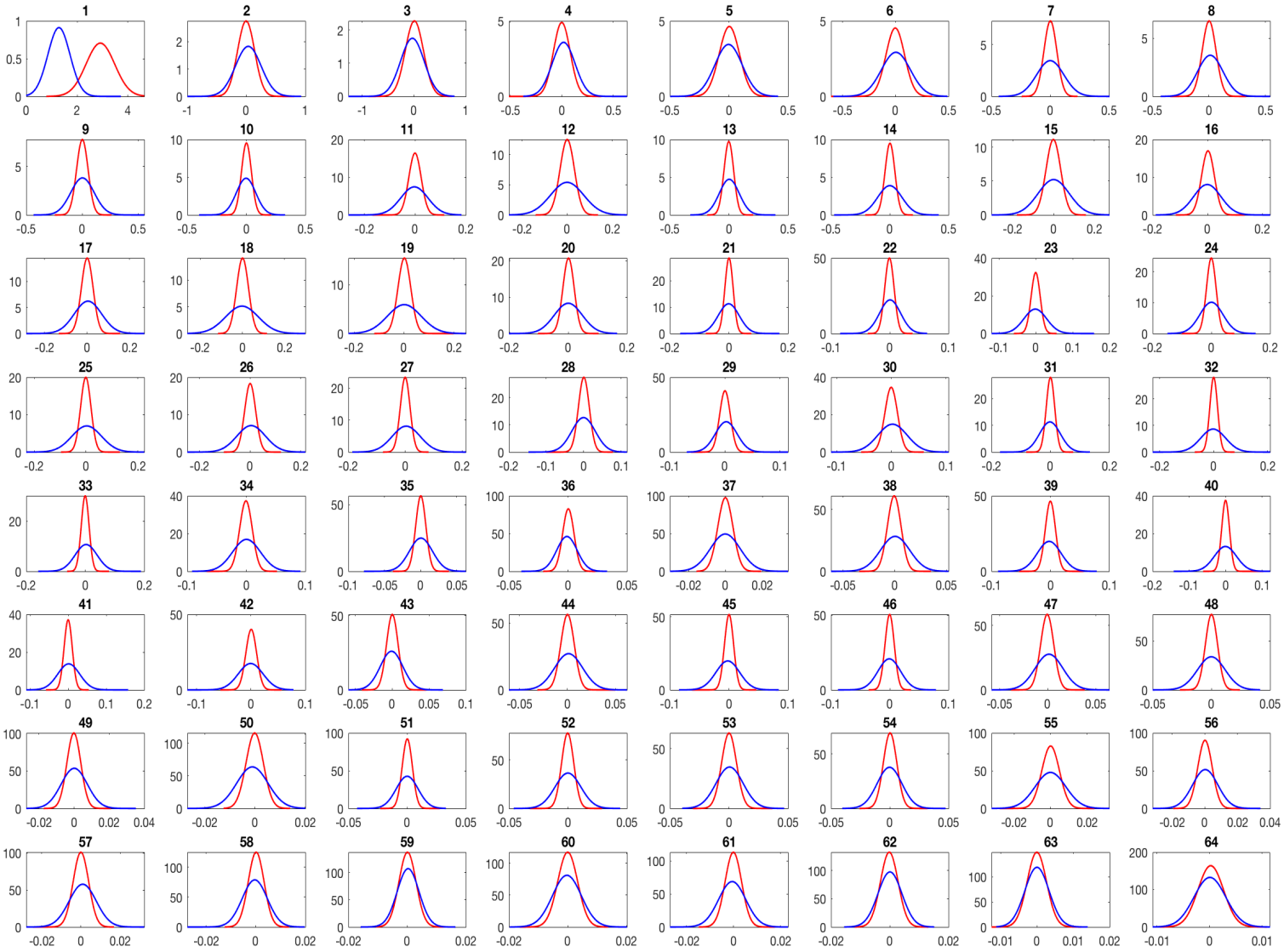
We can calculate the marginal densities for the two classes by the equations

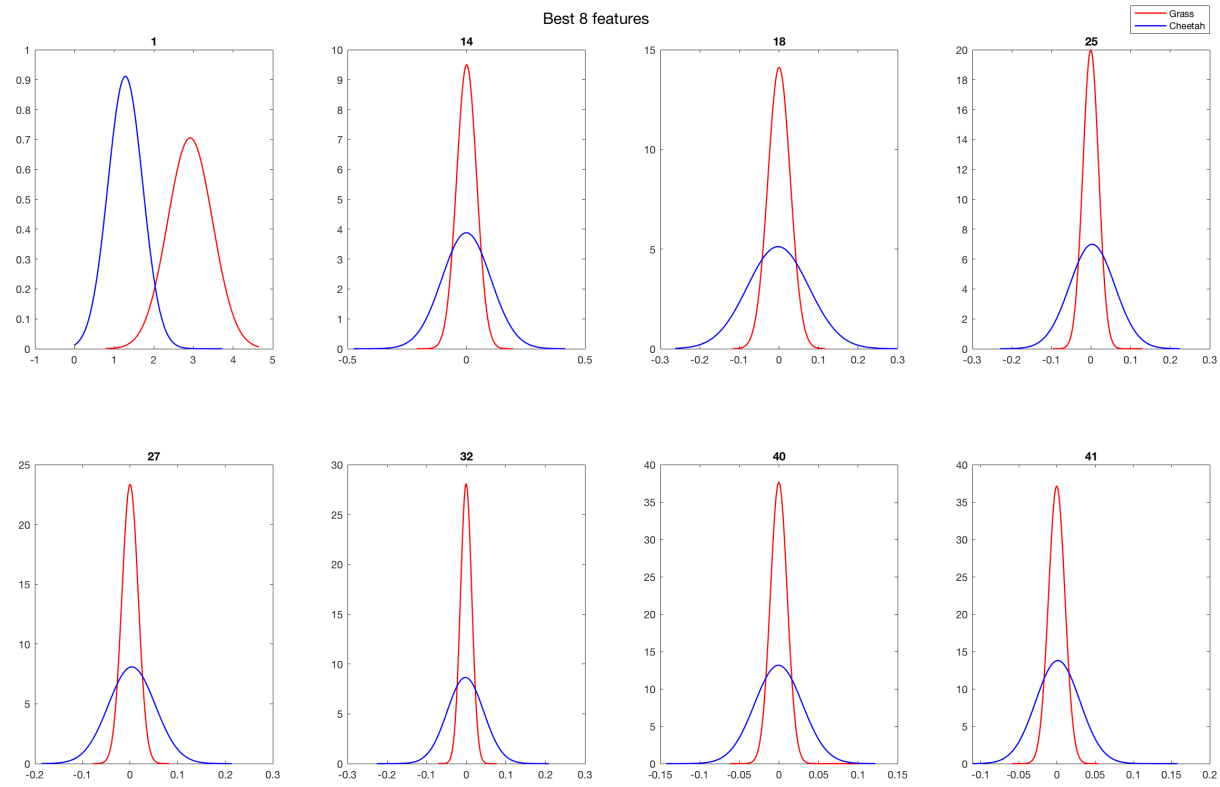
$$P_{x_k|y}(x_k|\text{grass/cheetah}) = \frac{1}{\sqrt{(2\pi)\sigma^2}}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}$$

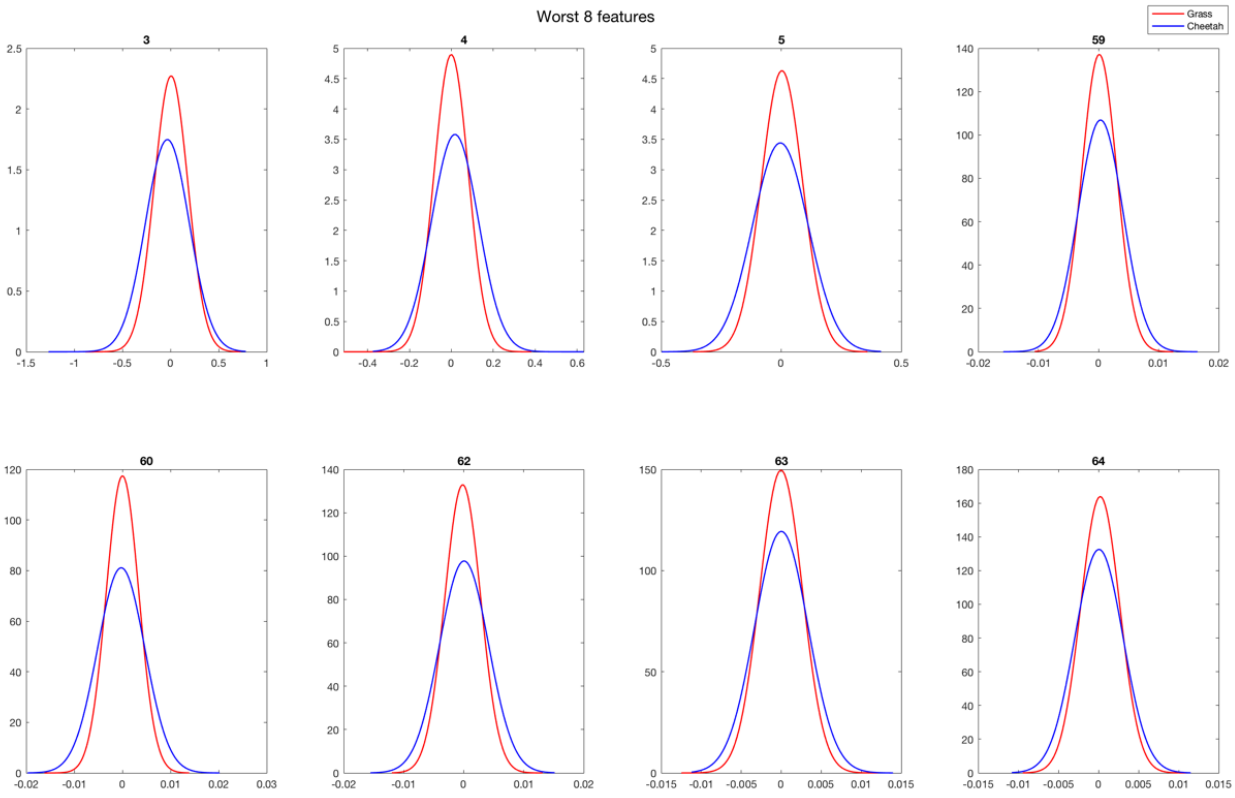
$$\mu_i = \frac{1}{n}\sum_j x_j^{(i)}$$

$$\Sigma_i = \frac{1}{n-1}\sum_j (x_j^{(i)} - \mu_i)(x_j^{(i)} - \mu_i)^T$$

Therefore, we can plot the marginal probability density functions for each feature. Figures are shown below. I think the best 8 features are 1, 14, 18, 25, 27, 32, 40, 41. The worst 8 features are 3, 4, 5, 59, 60, 62, 63, 64.




Best 8 features
Grass
Cheetah
1
14
18
25
27
32
40
41


Worst 8 features
Grass
Cheetah
3
4
5
59
60
62
63
64

c) Compute the Bayesian decision rule and classify the locations of the cheetah image using i) the 64-dimensional Gaussians, and ii) the 8-dimensional Gaussians associated with the best 8 features. For the two cases, plot the classification masks and compute the probability of error by comparing with cheetah mask.bmp. Can you explain the results?

Answer:

To segment the cheetah.bmp, we need to break it into 8*8 blocks. Load the cheetah image, we got a 255*270 matrix. I added 7 columns ones after and 7 rows ones below the matrix to make sure every pixel can have an 8*8 block. Each block represented the top-left pixel. Then, I computed DCT and ordered coefficients with zig-zag scan.

According to the Gaussian classifier, Bayesian decision rule can be written as

$$i^*(\mathrm{x}) = \arg\min[\, d_i\,(x,\, \mu_i) + \alpha_i]$$

with

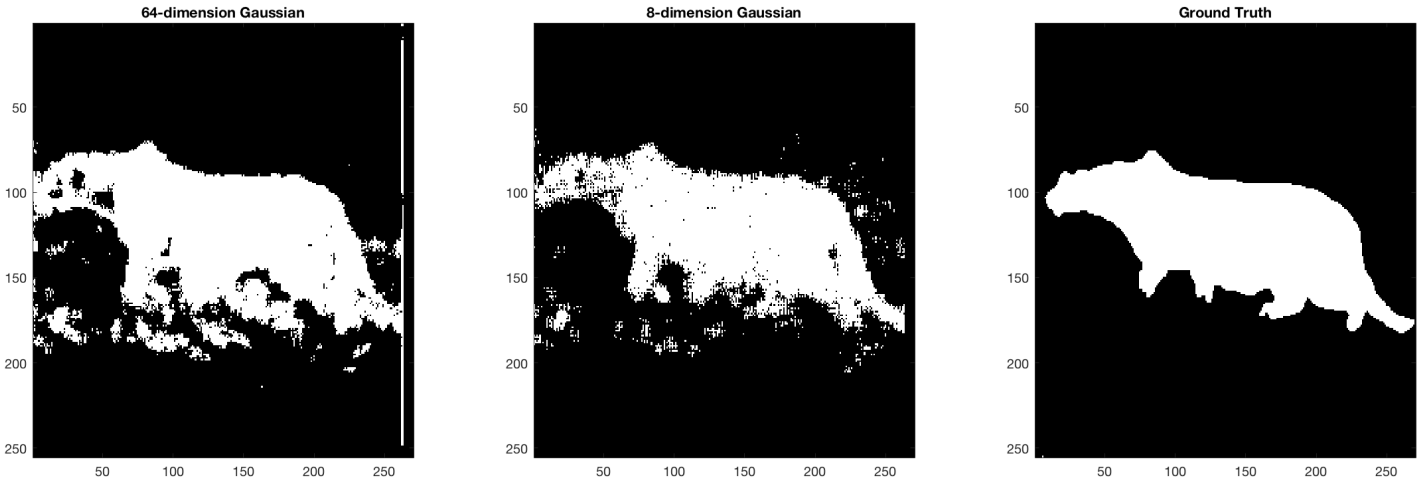
$$d_i\,(x,\mathrm{y}) = (x-y)^T \sum_i^{-1} (x-y)$$
$$\alpha_i\ = log(2\pi)^d \Sigma_i - 2logP_y(i)$$

From the training data, we can get the means and covariances for 64-dimensional Gaussians and 8-dimensional Gaussians. x is the ordered vector derived from each 8*8 block. If

$$(d_i\,(x,\, \mu_i) + \alpha_i)_{cheetah} \geq\ (d_i\,(x,\, \mu_i) + \alpha_i)_{grass}$$

we classify the pixels as “grass” and denoted by “0”. Else, we classify the pixels as “cheetah” and denoted by “1”.

Therefore, we can classify each pixel in the cheetah image into “cheetah” or “grass” by Bayesian decision rule using 64-dimensional Gaussians and 8-dimensional Gaussians associated with the best 8 features, respectively and get the segmented images. The segmented images are shown below with ground truth.



$$\text{For } 64-\text{dimensional Gaussians}$$

$$\text{Error(FG)} = \frac{\text{the number of "Cheetah" } pixels\ misclassified\ as\ "grass"}{\text{the number of "Cheetah" } pixels\ in\ ground\ truth\ image} = \frac{965}{13209} = 0.0731$$

$$\text{Error(BG)} = \frac{\text{the number of "}grass\text{" } pixels\ misclassified\ as \text{ "Cheetah"}}{\text{the number of "}grass\text{" } pixels\ in\ ground\ truth\ image} = \frac{5205}{55641} = 0.0935$$

$$\text{Error} = \text{Error(FG)} * \text{Py(cheetah)} + \text{Error(BG)} * \text{Py}(grass) = 0.0731 * 0.1919 + 0.0935 * 0.8081 = 0.0896$$

For $8-$dimensional Gaussians

$$\text{Error(FG)} = \frac{\text{the number of "Cheetah" } pixels\ misclassified\ as \text{ "}grass\text{"}}{\text{the number of "Cheetah" } pixels\ in\ ground\ truth\ image} = \frac{1480}{13209} = 0.1120$$

$$\text{Error(BG)} = \frac{\text{the number of "}grass\text{" } pixels\ misclassified\ as \text{ "Cheetah"}}{\text{the number of "}grass\text{" } pixels\ in\ ground\ truth\ image} = \frac{2413}{55641} = 0.0434$$

$$\text{Error} = \text{Error(FG)} * \text{Py(cheetah)} + \text{Error(BG)} * \text{Py}(grass) = 0.1120 * 0.1919 + 0.0434 * 0.8081 = 0.0566$$

Both results are better than the result in Assignment one. We could see that 8-dimensional Gaussians produced less errors than 64-dimensional Gaussians. This is because we picked up the best 8 features which are the most distinctive features between grass and cheetah. Therefore, we got a pretty good result. However, when we used the 64-dimensional Gaussians, some features can be really similar between grass and cheetah which may confuse the system, then lead to a worse result.

Matlab Code :

```
%%
struct = open('TrainingSamplesDCT_8_new.mat');
data = struct2cell(struct);
TrainsampleDCT_FG =cell2mat(data(1,:));
TrainsampleDCT_BG =cell2mat(data(2,:));
[FG_m, FG_n] = size(TrainsampleDCT_FG);
[BG_m, BG_n] = size(TrainsampleDCT_BG);

%% Prior
FG = zeros(1,FG_m);
BG = ones(1,BG_m);
total = [FG,BG];
hist = histogram(total,'Normalization','probability');
xlim([-2,3])
xticks([0.1 1.1])
xticklabels({'FG__cheetah','BG__grass'})
ylabel('Prior Probability Py')
title('Cheetah/Grass histograms')

Py_cheetah = FG_m/(FG_m+BG_m);
Py_grass = BG_m/(FG_m+BG_m);
```

```
% mean and covariance 64
mu_b = sum(TrainsampleDCT_BG,1)/BG_m;
mu_f = sum(TrainsampleDCT_FG,1)/FG_m;
cof_b = TrainsampleDCT_BG - mu_b;
cof_f = TrainsampleDCT_FG - mu_f;
cov_b = cof_b.'*cof_b/(BG_m-1);
cov_f = cof_f.'*cof_f/(FG_m-1);
%% Px|y
% 64 plot of two classes
figure;
for i = 1:64
    subplot(8,8,i)

    [f,x] = ksdensity(TrainsampleDCT_BG(:,i));%get a propriate range x
    mu = mu_b(i);
    sigma2 = cov_b(i,i);
    y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
    plot(x,y,'r','linewidth',1.2)

    hold on
    [f,x] = ksdensity(TrainsampleDCT_FG(:,i));%get a propriate range x
    mu = mu_f(i);
    sigma2 = cov_f(i,i);
    y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
    plot(x,y,'b','linewidth',1.2)
    title(num2str(i))
end
legend('Grass','Cheetah');

%best 8
best8 = [1,14,18,25,27,32,40,41];
figure;
sgtitle('Best 8 features');
n=1;
for i = 1:64
    if ~isempty(find(i==best8, 1))
        %grass
        subplot(2,4,n)
        n = n+1;
        [f,x] = ksdensity(TrainsampleDCT_BG(:,i));%get a propriate range x
        mu = mu_b(i);
        sigma2 = cov_b(i,i);
        y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
        plot(x,y,'r','linewidth',1.2)
```

```
        hold on
        %cheetah
        [f,x] = ksdensity(TrainsampleDCT_FG(:,i));%get a propriate range x
        mu = mu_f(i);
        sigma2 = cov_f(i,i);
        y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
        plot(x,y,'b','linewidth',1.2)
        title(num2str(i))
    end
end
legend('Grass','Cheetah');


%worst 8
worst8 = [3,4,5,59,60,62,63,64];
figure;
sgtitle('Worst 8 features');
n=1;
for i = 1:64
    if ~isempty(find(i==worst8, 1))
        %grass
        subplot(2,4,n)
        n = n+1;
        [f,x] = ksdensity(TrainsampleDCT_BG(:,i));%get a propriate range x
        mu = mu_b(i);
        sigma2 = cov_b(i,i);
        y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
        plot(x,y,'r','linewidth',1.2)

        hold on
        %cheetah
        [f,x] = ksdensity(TrainsampleDCT_BG(:,i));%get a propriate range x
        mu = mu_f(i);
        sigma2 = cov_f(i,i);
        y=1/((sqrt(2*pi*sigma2)))*exp(-((x-mu).^2)/(2*sigma2));
        plot(x,y,'b','linewidth',1.2)
        title(num2str(i))
    end
end
legend('Grass','Cheetah');
```

```
%%
indexPatten = load('Zig-Zag Pattern.txt') +1;
indexPatten = reshape(indexPatten,1,64);
testData = im2double(imread('cheetah.bmp'));
[t_m, t_n] = size(testData);
segImg = zeros(t_m ,t_n);
testData(t_m:t_m+7,t_n:t_n+7) = 1;

%64-dimension Gaussian
for i = 1:t_m - 7
   for j = 1:t_n - 7
       tempMatrix = testData(i:i+7, j:j+7);
       tempMatrix = dct2(tempMatrix);
       tempArray = reshape(tempMatrix,1,64);
       arrayWithIndex = [indexPatten.',tempArray.'];
       sortArray = sortrows(arrayWithIndex,1);
       x = sortArray(:,2);
       result = GC(x, mu_f, cov_f, Py_cheetah, mu_b, cov_b, Py_grass);
       segImg(i,j) = double(result);
   end
end
figure;
subplot(1,3,1)
imagesc(segImg);
colormap(gray(255));
title('64-dimension Gaussian');
[mis_ftob_64,mis_btof_64,f_num,b_num] = err(segImg);
error_ftob_64 = mis_ftob_64 / f_num;
error_btof_64 = mis_btof_64 / b_num;
Error_64 = error_btof_64 * Py_grass + error_ftob_64 * Py_cheetah;

%8-dimension Gaussian
cov_f = cov_f(best8,best8);
cov_b = cov_b(best8,best8);
mu_f = mu_f(best8);
mu_b = mu_b(best8);

for i = 1:t_m - 7
   for j = 1:t_n - 7
       tempMatrix = testData(i:i+7, j:j+7);
       tempMatrix = dct2(tempMatrix);
       tempArray = reshape(tempMatrix,1,64);
       arrayWithIndex = [indexPatten.',tempArray.'];
       sortArray = sortrows(arrayWithIndex,1);
       x = sortArray(best8,2);
```

```
        result = GC(x, mu_f, cov_f, Py_cheetah, mu_b, cov_b, Py_grass);
        segImg(i,j) = double(result);
    end
end
subplot(1,3,2)
imagesc(segImg);
colormap(gray(255));
title('8-dimension Gaussian');
[mis_ftob_8,mis_btof_8,f_num,b_num] = err(segImg);
error_ftob_8 = mis_ftob_8 / f_num;
error_btof_8 = mis_btof_8 / b_num;
Error_8 = error_btof_8 * Py_grass + error_ftob_8 * Py_cheetah;


%% Error
maskImg=im2double(imread('cheetah_mask.bmp'));
subplot(1,3,3)
imagesc(maskImg);
colormap(gray(255));
title('Ground Truth');

%error
function [mis_ftob,mis_btof,f_num,b_num] = err(segImg)
    maskImg=im2double(imread('cheetah_mask.bmp'));
    maskImg_data = maskImg(:);
    f_num = sum (maskImg_data);
    b_num = length(maskImg_data) - f_num;
    segImg_data = segImg(:);
    mis_ftob = 0;
    mis_btof = 0;
    for i = 1:length(segImg_data)
        if segImg_data(i) == 0 && segImg_data(i) ~= maskImg_data(i)
            mis_ftob = mis_ftob + 1;
        elseif segImg_data(i) == 1 && segImg_data(i) ~= maskImg_data(i)
            mis_btof = mis_btof + 1;
        end
    end



end

%classification
function result = GC(x, mu_f, sigma_f, Py_f, mu_b, sigma_b, Py_b)
    d = size(sigma_f,1);
```

```
    d_f = (x.' - mu_f)*inv(sigma_f)*(x-mu_f.');
    a_f = log((2*pi)^d*det(sigma_f)) - 2*log(Py_f);
    d_b = (x.' - mu_b)*inv(sigma_b)*(x-mu_b.');
    a_b = log((2*pi)^d*det(sigma_b)) - 2*log(Py_b);
    if d_f+a_f >= d_b+a_b
        result = 0;
    else
        result = 1;
    end
end
```