Jiaming Lai
A53314574

Assignment #2

Computer Problem Solution

a) **Solution**:
Using the results of problem 2, the maximum likelihood estimate for the prior probabilities is following:

$$P_Y(cheetah) = N_{FG}/(N_{FG} + N_{BG}) = 0.1919$$

$$P_Y(grass) = N_{BG}/(N_{FG} + N_{BG}) = 0.8081$$

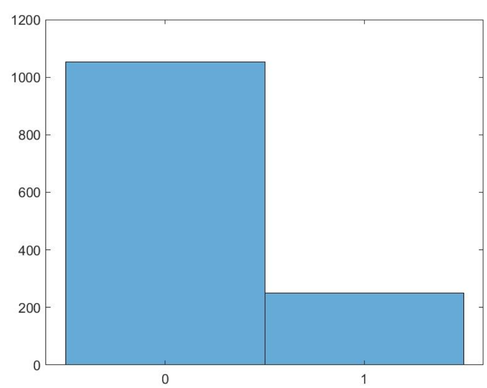
where

N_{BG} is the number of grass samples

N_{FG} is the number of cheetah samples

These estimates are actually the same as the estimates that I obtained in Homework#1. It implies that the prior probabilities could be estimated by using proportion of each states in the samples if we have enough or big number of samples.

The histogram is shown below. Mark cheetah to 1 and grass to 0. We could use the histogram to estimate $P_Y(cheetah)$ and $P_Y(grass)$. The result is the same as ML estimates.



b) **Solution**:
The marginal densities for two classes

$$P_{x_k|Y}(x_k|cheetah) \ and \ P_{x_k|Y}(x_k|grass), \ k = 1, 2, \ldots, 64$$

are shown below on Figure 1. The blue line represents $P_{x_k|Y}(x_k|cheetah)$ and the red line represents $P_{x_k|Y}(x_k|grass)$.

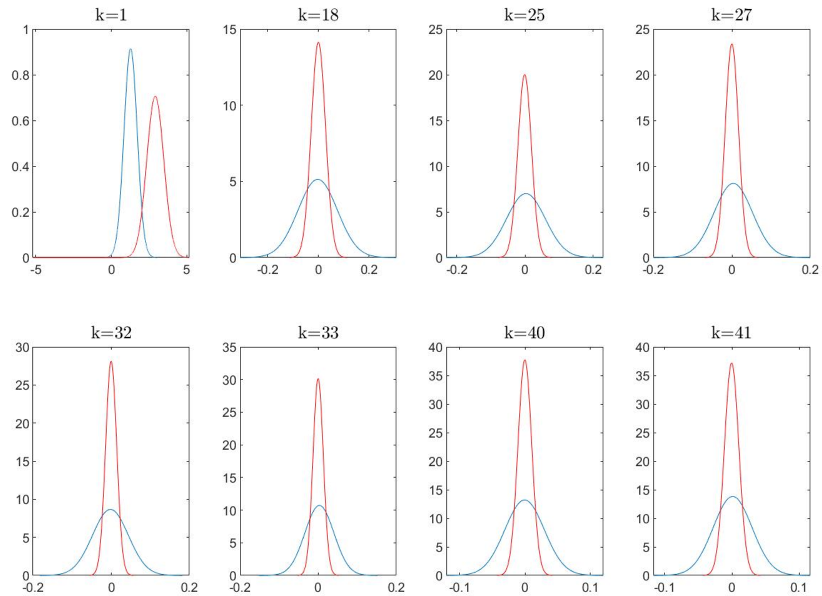
Figure 1: Marginal densities for two classes

By visual inspection, the best 8 features and worst 8 features are:

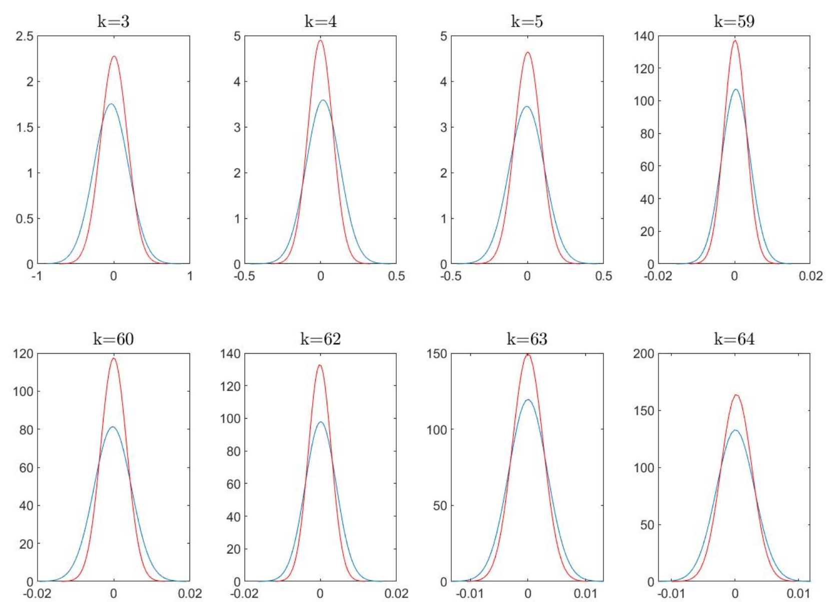
best 8 features:[1,18,25,27,32,33,40,41]

worst 8 features:[3,4,5,59,60,62,63,64]

The plots of the marginal densities for the best-8 and worst-8 features are shown below:



(a) Best 8 features



(b) Worst 8 features

Figure 2: Marginal densities for the best-8 and worst-8 features

c) **Solution**:
By using Bayesian decision rule, we can compare the proportion of conditional probabilities of two classes, compare the result with the threshold and mask the top left corner of the 8*8 block as 1, regarding this pixel belongs to cheetah. Otherwise, we mask 0.

$$\frac{P_{X|Y}(x|cheetah)}{P_{X|Y}(x|grass)} > T = \frac{P_Y(grass)}{P_Y(cheetah)}$$

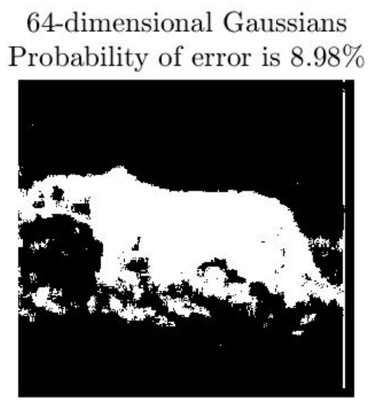
where

$P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ are conditional probabilities of two classes.
$P_Y(cheetah)$ and $P_Y(grass)$ are the ML estimates we get from training data.
T is the threshold.

$P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ is calculated using the following equation

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

The classification results using 64-dimensional Gaussians and 8-dimensional Gaussians associated with the best 8 features are shown below. Compare it with the ground truth provided in image **cheetah mask.bmp** and compute the probability of error. The probability of error is also shown in the figure below.



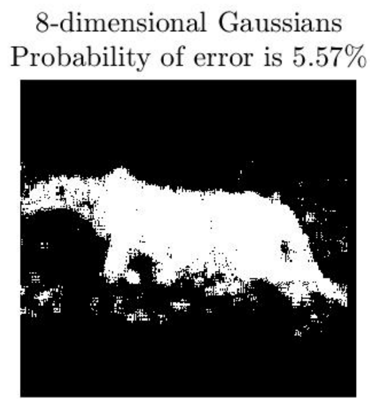


Figure 3: Classification results

Obviously, the classification performance of 8-dimensional Gaussians associated with the best 8 features is better than the 64-dimensional Gaussians. My explanation is that while we add more features to the best 8 features, the probability density distribution of $P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ will get closer, bacause some features' marginal densities are very similiar, just like the worst 8 features we shown above. Thus, it makes Bayesian decision rule based classification perform worse.

Appendix

The following is the Matlab code.

0.1 HW2_solution.m

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem (a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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));
%Plot the histogram
his = [ones(size(train_FG,1),1);zeros(size(train_BG,1),1)];
h1 = histogram(his);
set(gca,'XTick',[0:1:2]);
saveas(gcf, ['Images/histogram.jpg']);
close(gcf);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem (b)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculate the mean of every features when cheetah
%mean_ch is the mean
mean_ch = fun_mean(train_FG);
%Calculate the mean of every features when grass
%mean_gr is the mean
mean_gr = fun_mean(train_BG);

%Calculate the covariance matrix of cheetah
cov_ch = fun_cov(train_FG,mean_ch);
```

```
%Calculate the covariance matrix of grass
cov_gr = fun_cov(train_BG,mean_gr);


%std_gr = std(train_BG,0,1);

%Plot the 64-plots
for j=1:1:4
    position = 1; %Define the subplot row index
    for i=(j-1)*16+1:1:(j-1)*16+16
        figure(j);
        subplot(4,4,position);
        x=-(mean_ch(i)+4*sqrt(cov_ch(i,i))):0.0005:(mean_ch(i)+4*sqrt(
   cov_ch(i,i)));
        y=fun_gaussian(x,mean_ch(i),sqrt(cov_ch(i,i)));
        %Plot the cheetah line
        plot(x,y);
        hold on
        x=-(mean_gr(i)+4*sqrt(cov_gr(i,i))):0.0005:(mean_gr(i)+4*sqrt(
   cov_gr(i,i)));
        y=fun_gaussian(x,mean_gr(i),sqrt(cov_gr(i,i)));
        %Plot the grass line and mark it as red
        plot(x,y,'r');
        position = position + 1;
        title({['k=',num2str(i)]},'Fontsize',12,'interpreter','latex');
    end
    set(gcf,'Position',[400,100,900,600]);
    %Save the images
    saveas(gcf, ['Images/subplot',num2str(j),'.jpg']);
    close(gcf);
end

best = [1,18,25,27,32,33,40,41];
worst = [3,4,5,59,60,62,63,64];

%Plot the best 8 features
position = 1;
for i=best
    subplot(2,4,position);
    x=-(mean_ch(i)+4*sqrt(cov_ch(i,i))):0.0005:(mean_ch(i)+4*sqrt(cov_ch(i
   ,i)));
    y=fun_gaussian(x,mean_ch(i),sqrt(cov_ch(i,i)));
    %Plot the cheetah line
    plot(x,y);
    hold on
    x=-(mean_gr(i)+4*sqrt(cov_gr(i,i))):0.0005:(mean_gr(i)+4*sqrt(cov_gr(i
   ,i)));
    y=fun_gaussian(x,mean_gr(i),sqrt(cov_gr(i,i)));
    %Plot the grass line and mark it as red
    plot(x,y,'r');
```

```
    position = position + 1;
    title({['k=',num2str(i)]},'Fontsize',12,'interpreter','latex');
end
set(gcf,'Position',[400,100,900,600]);
%Save the images
saveas(gcf, ['Images/subplot_best8features.jpg']);
close(gcf);

%Plot the worst 8 features
position = 1;
for i=worst
    subplot(2,4,position);
    x=-(mean_ch(i)+4*sqrt(cov_ch(i,i))):0.0005:(mean_ch(i)+4*sqrt(cov_ch(i
   ,i)));
    y=fun_gaussian(x,mean_ch(i),sqrt(cov_ch(i,i)));
    %Plot the cheetah line
    plot(x,y);
    hold on
    x=-(mean_gr(i)+4*sqrt(cov_gr(i,i))):0.0005:(mean_gr(i)+4*sqrt(cov_gr(i
   ,i)));
    y=fun_gaussian(x,mean_gr(i),sqrt(cov_gr(i,i)));
    %Plot the grass line and mark it as red
    plot(x,y,'r');
    position = position + 1;
    title({['k=',num2str(i)]},'Fontsize',12,'interpreter','latex');
end
set(gcf,'Position',[400,100,900,600]);
%Save the images
saveas(gcf, ['Images/subplot_worst8features.jpg']);
close(gcf);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem (c)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The 64-dimensional Gaussians
% %Calculate the covariance matrix of cheetah
% cov_ch = fun_cov(train_FG,mean_ch);
% %Calculate the covariance matrix of grass
% cov_gr = fun_cov(train_BG,mean_gr);

%Read original image
I = imread('dataset/cheetah.bmp');
I = im2double(I);
%Define the loop numbers
loop_row = size(I,1) - 8 + 1;
loop_column = size(I,2) - 8 + 1;
%Caculate the threshold
T = P_BG / P_FG;

mask_64 = zeros(size(I));
```

```
%Read the Zig-Zag file
position_ref = load('dataset/Zig-Zag Pattern.txt');
%Define the array for saving DCT coeffiences according to Zig-Zag
DCT_coeffience = zeros([1,64]);

for i=1:1:loop_row
    for j=1:1:loop_column
        block = I(i:i+7,j:j+7);
        DCT_block = dct2(block);
        %Map DCT_block matrix to array according Zig-Zag
        for row=1:1:8
            for column=1:1:8
                DCT_coeffience(1,position_ref(row,column)+1)=DCT_block(row
    ,column);
            end
        end
        P_x_FG = fun_mvgaussian(DCT_coeffience,mean_ch,cov_ch);
        P_x_BG = fun_mvgaussian(DCT_coeffience,mean_gr,cov_gr);
        if P_x_FG/P_x_BG > T
            mask_64(i,j) = 1;
        end
    end
end

%The best 8 features
%Calculate the covariance matrix of cheetah
cov_ch = fun_cov(train_FG(:,best),mean_ch(:,best));
%Calculate the covariance matrix of grass
cov_gr = fun_cov(train_BG(:,best),mean_gr(:,best));

mask_8 = zeros(size(I));
%Define the array for saving DCT coeffiences according to Zig-Zag
DCT_coeffience = zeros([1,64]);

for i=1:1:loop_row
    for j=1:1:loop_column
        block = I(i:i+7,j:j+7);
        DCT_block = dct2(block);
        %Map DCT_block matrix to array according Zig-Zag
        for row=1:1:8
            for column=1:1:8
                DCT_coeffience(1,position_ref(row,column)+1)=DCT_block(row
    ,column);
            end
        end
        P_x_FG = fun_mvgaussian(DCT_coeffience(best),mean_ch(best),cov_ch)
    ;
        P_x_BG = fun_mvgaussian(DCT_coeffience(best),mean_gr(best),cov_gr)
    ;
        if P_x_FG/P_x_BG > T
```

```
                mask_8(i,j) = 1;
            end
        end
end

%Read the mask file
I = imread('dataset/cheetah_mask.bmp');
I = im2double(I);
subplot(1,2,1)
imshow(mask_64);
%Calculate the probability of error
error = length(find((mask_64-I)~=0)) / (size(I,1) * size(I,2));
title({['64-dimensional Gaussians'];['Probability of error is ',num2str(error*100,'%.2f'),'\%']},'Fontsize',12,'interpreter','latex');
subplot(1,2,2)
imshow(mask_8);
%Calculate the probability of error
error = length(find((mask_8-I)~=0)) / (size(I,1) * size(I,2));
title({['8-dimensional Gaussians'];['Probability of error is ',num2str(error*100,'%.2f'),'\%']},'Fontsize',12,'interpreter','latex');
%Save the image
saveas(gcf, ['Images/segmentation.jpg']);
close(gcf);
```

0.2 fun_mean.m

The function for calculating ML mean.

```
function [mean] = fun_mean(data)
%This function is the maximum likelihood estimation of mean

N = size(data,1);
A = ones(1,N);
mean = A * data ./ N;

end
```

0.3 fun_cov.m

The function for calculating covariance matrix and ML variance.

```
function [cov] = fun_cov(data,data_mean)
%This function is the maximum likelihood estimation of covariance matrix

N = size(data,1);
cov = (data-data_mean)'*(data-data_mean)./N;

end
```

0.4 fun_gaussian.m

```
function [y] = fun_gaussian(x,mean,variance)
%This function is for Gaussian

y = exp(-(x-mean).^2/2/variance^2)./variance./sqrt(2*pi);

end
```

0.5 fun_mvgaussian.m

```
function [y] = fun_mvgaussian(x,mean,cov)
%This function is for MV Gaussian

d = size(x,2);
y = 1/sqrt((2*pi)^d*det(cov))*exp(-(x-mean)*cov^-1*(x-mean)'/2);

end
```