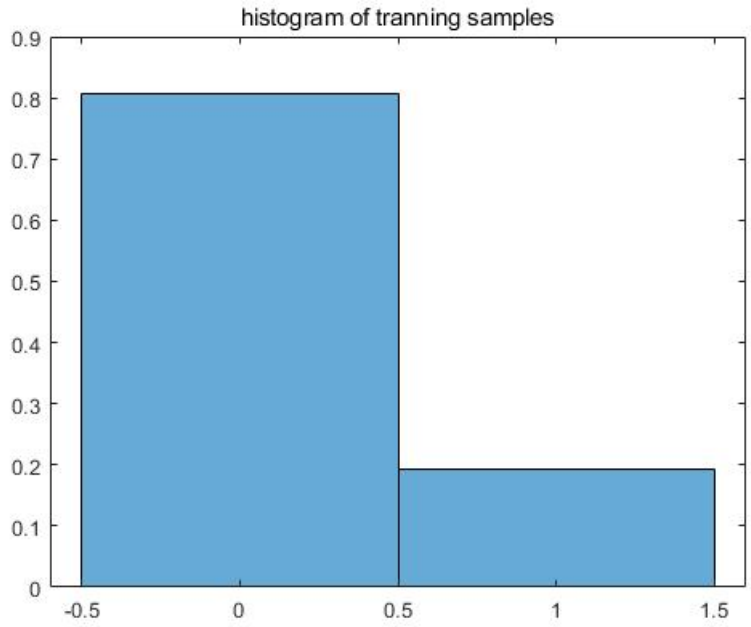
[ECE 271A] Homework Set 2

Problem 6

Solution

a) The histogram of 2 training sets is shown below:



According to the results of Problem2, we have the maximum likelihood of prior probability is

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

of which c_j is the frequency of the sample of observations, n is the total of times of observations.

in this case, we can consider that each row represents one observation of cheetah or grass, the total number of rows in each training sets to be $N_{cheetah}$ and N_{grass}, that is to say

$$P_Y(cheetah) = \frac{N_{cheetah}}{N_{cheetah} + N_{grass}}$$

with the constrain of

$$\Sigma_j \pi_j^* = 1$$

in this case, there are only to states, which are $Y = \{cheetah, grass\}$. We can imply that

$$P_Y(cheetah) + P_Y(grass) = 1$$

The *prior probabilities* $P_Y(i), i \in (cheetah, grass)$ is the same as <Homework Set 1>.

b) Under Gaussian assumption, consider each position X_k separately, density of all $x_i \in X_k$ in the training sets should follow

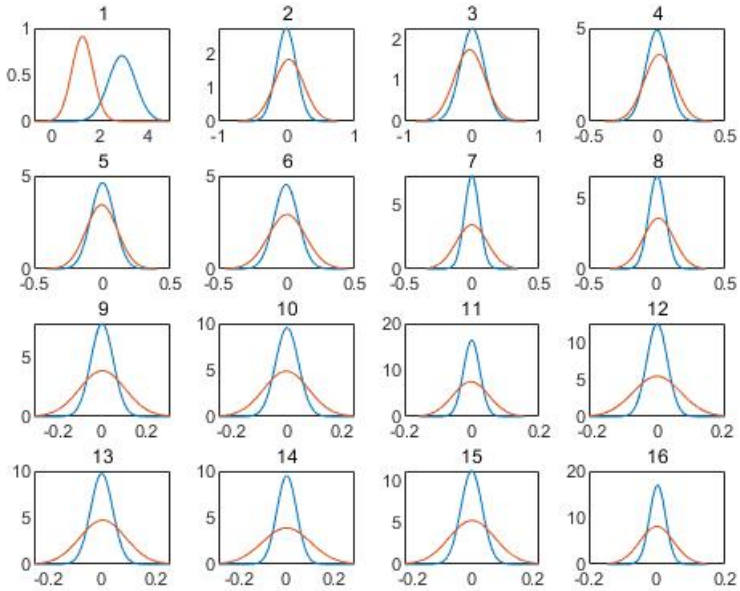
$$G(x, \mu_k, \sigma_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu_k)^2}{2\sigma^2}}$$

$$\mu_k = \frac{1}{n}\Sigma_{i=1}^{n} x_i$$

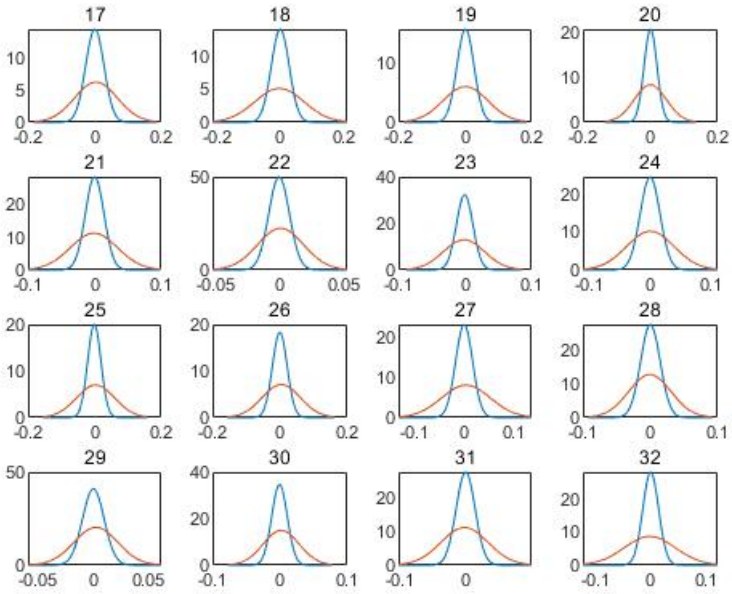
$$\sigma_k = \frac{1}{n}\Sigma_{i=1}^{n} (xi - \mu_k)^2$$

the Gaussian distribution plot of x^{th} value of DCT of *Cheetah's training set* and *Background's training set* are calculated and shown as follow:

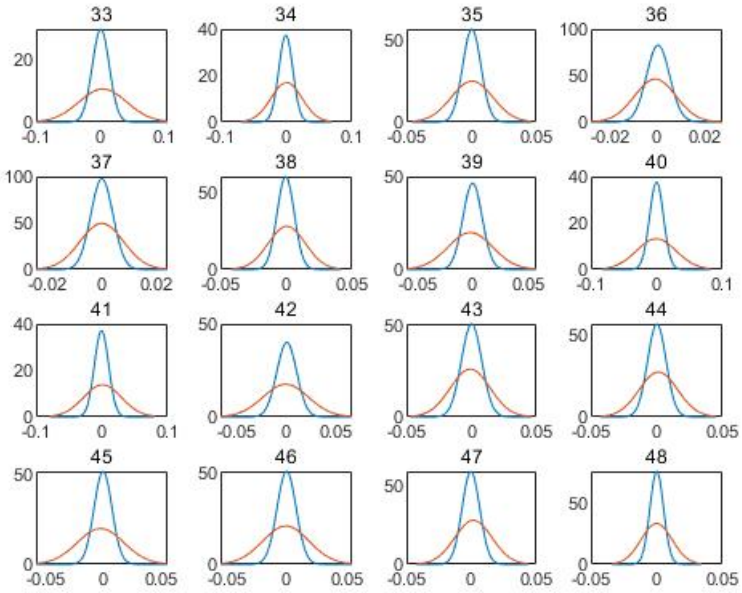
Gaussian Distribution of x = 1,...,16



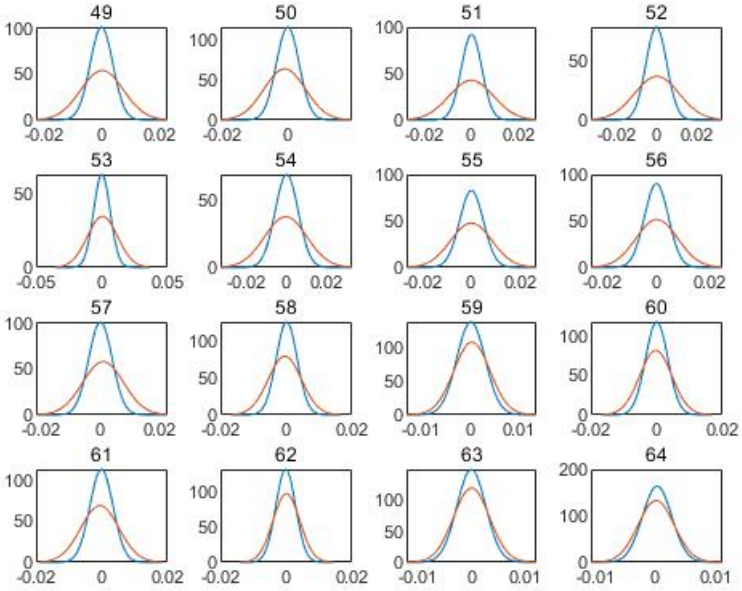
Gaussian Distribution of x = 17,...,32



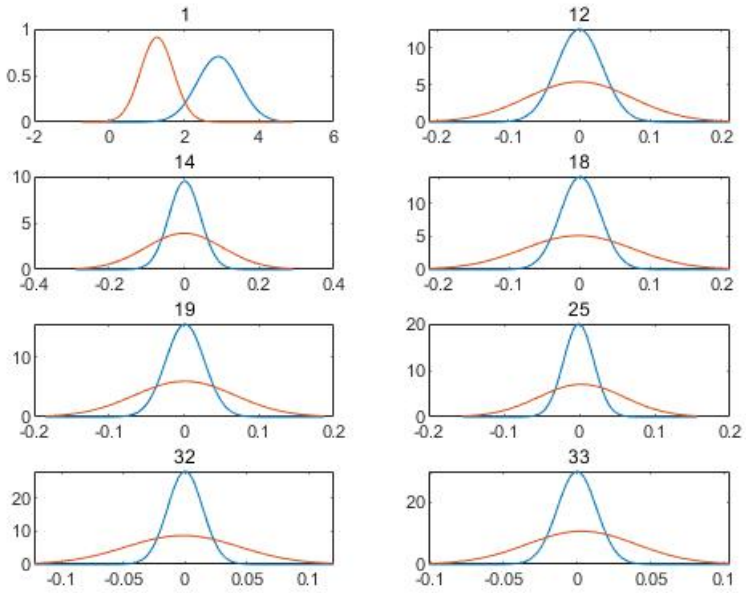
Gaussian Distribution of x = 33,...,48



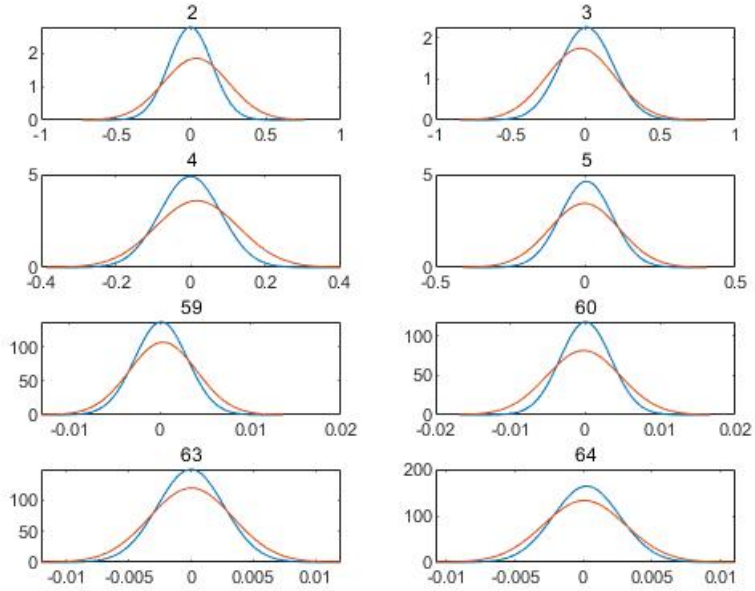
Gaussian Distribution of x = 49,...,64



Among these Gaussian Distribution curves, by personal visual inspection, the best 8 features (ordered by feature's number) are:



and the worst 8 features (ordered by feature's number) are:



c) If we select all 64 features of X to be involved in classification process, for each vector observation $X = [x_k], (k = 1, 2, \ldots, 64)$, the likelihood function is

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

in which,

$$\mu = Exp(X)$$
$$\Sigma = E\left[(X - E(X))(X - E(X))^T\right]$$

Then applying the Bayes Rule, if

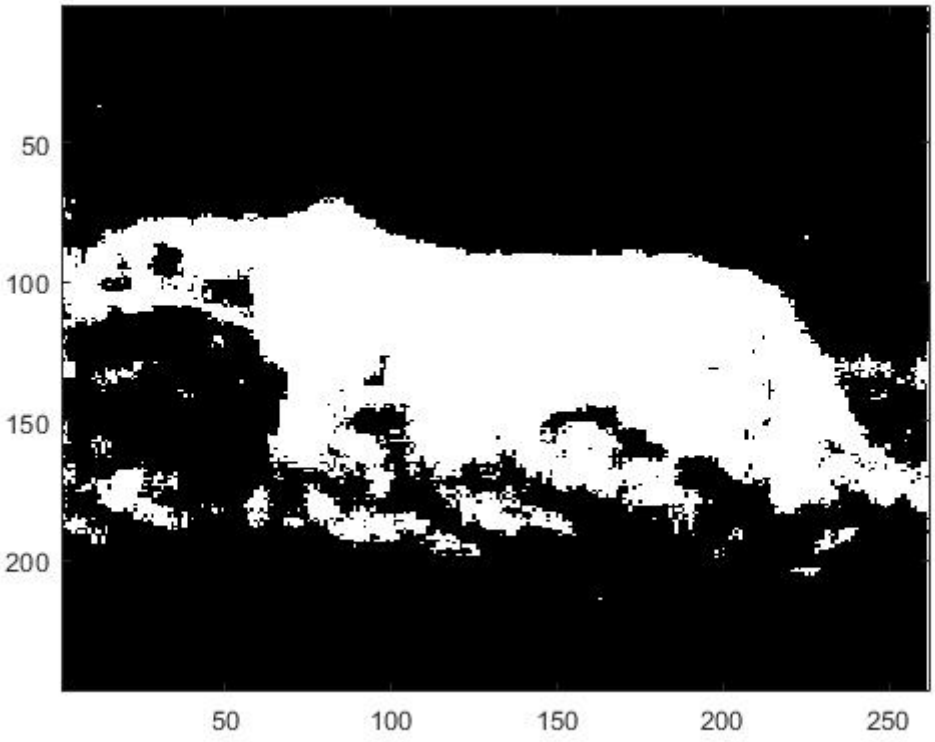
$$\frac{P_{X|Y}(x|cheetah)P_Y(cheetah)}{P_{X|Y}(x|grass)P_Y(grass)} > 1$$

then infer the pixel belongs to cheetah-class rather than grass-class. The other cases, let the pixel belongs to grass-class. And the error rate can be calculated as

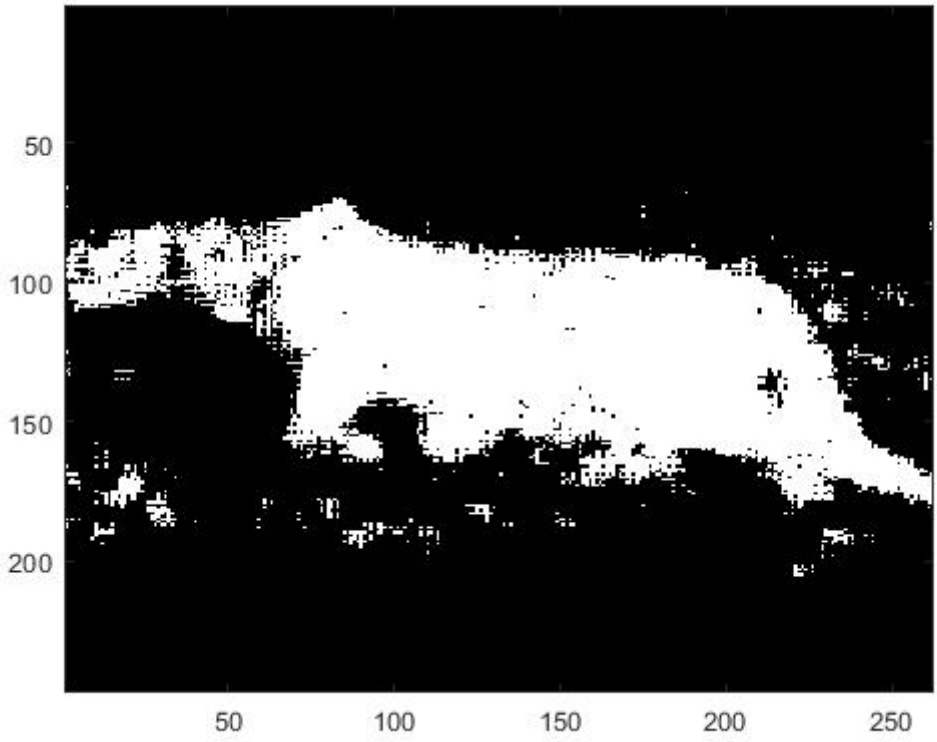
$$ErrorRate = Exp(\Sigma|v - v_{mask}|)$$

in which, v is the estimated value of each pixel, v_{mask} is treated as the true results of each pixel. $v, v_{mask} \in \{0, 1\}$.

The result of 64-dimensional Gaussians classification is shown as below, with an error rate of 0.085955



The result of 8-dimensional Gaussians classification is shown as below, with an error rate of 0.052389



To explain the results that "with fewer features, the error rate declines", personally thinking, there are possibly some factors that cause this situation.

1) Overfitting. 64-dimension may contain over-sufficient features for this 0,1-classification. As a result of this, some Gaussian features from the training sets are not quite fit the test set.

2) Insufficient training samples. Maybe the exist training sample cannot absolutely represent all the properties of Gaussian features calculated in the test set, and some features may cause higher risks than the chosen "best 8 features", because their distribution mainly cover each other.

3) Computing error. The 64-dimensional Gaussian calculation, when calculating the inverse of covariance matrices, loses its numerical precision, which may cause some pixel being misclassified. And it is less severe when calculate the inverse of a 8x8 dimensional matrix.

Appendix

loadTS_DCT8.m

```
function [BG, FG] = loadTS_DCT8(dtype,text)
    load(dtype,text);
    BG = TrainsampleDCT_BG;
    FG = TrainsampleDCT_FG;
end
```

sum_y.m

```
function a = sum_y(matrix, dim)
    if dim == 1
        a = matrix*ones([size(matrix,2),1]);
    elseif dim == 2
        a = ones([1,size(matrix,1)])*matrix;
    end
end
```

GaussianCalc.m

```
function [mu, sigma_2] = GaussianCalc(samples)
    mu = mean(samples,1);
    sigma_2 = samples(1,:)
end
```

MLpropsCalc.m

```
function [mu, sigma] = MLpropsCalc(samples)
    mu = sum_y(samples,2)/size(samples,1);
    sigma_2 = sum_y((samples-mu).*(samples-mu),2)./size(samples,1);
    sigma = sqrt(sigma_2);
end
```

GaussPlot.m

```
function GaussPlot(mu1,sig1,mu2,sig2,i)
    subi = mod(i,16);
    if (subi== 0)
        subi = 16;
    end
    subplot(4,4,subi);
    xmin = min(mu1,mu2)-2*(sig1+sig2);
    xmax = max(mu1,mu2)+2*(sig1+sig2);
    x = xmin:(xmax-xmin)/100:xmax;
    y1 = (sqrt(2*pi)*sig1).^(-1) * exp(-(x-mu1).^2/(2*sig1*sig1));
    y2 = (sqrt(2*pi)*sig2).^(-1) * exp(-(x-mu2).^2/(2*sig2*sig2));
    plot(x,y1);
    hold on
    plot(x,y2);
    title(i);
end
```

ndGauss.m

```
function g = ndGauss(x,mu,conv)
    if ~istall(x)
        x = x.';
    end
    if ~istall(mu)
        mu = mu.';
    end
    g = (2*pi)^(-size(x,1)/2)*(det(conv)^(-1/2))*exp(-((x-mu).')/(conv)*(x-
mu)*1/2);
end
```

hw2.6.m

```
% set an empty workspace
clc;
clear;

% load the Traning Samples
[BG, FG] = loadTS_DCT8('-mat','TrainingSamplesDCT_8_new.mat');

%Print the histogram
figure('Name','histogram of tranning samples');
histo =
histogram([zeros([1,size(BG,1)]),ones([1,size(FG,1)])],'Normalization','Probabil
ity');
title('histogram of tranning samples')

% Calculate the histogram of Prior Probablity of {cheetah, grass}
Py_cheetah = size(FG,1)/(size(FG,1)+size(BG,1));
Py_grass = 1 - Py_cheetah;

% Calculate the ML Properties
[BGmu,BGsigma] = MLpropsCalc(BG);
[FGmu,FGsigma] = MLpropsCalc(FG);

% plot the gaussian distribution
figcnt = 1;
figure(figcnt);
for i = 1:16
    GaussPlot(BGmu(i),BGsigma(i),FGmu(i),FGsigma(i),i);
end
figcnt = figcnt+1;
figure(figcnt);
for i = 17:32
    GaussPlot(BGmu(i),BGsigma(i),FGmu(i),FGsigma(i),i);
end
figcnt = figcnt+1;
figure(figcnt);
for i = 33:48
    GaussPlot(BGmu(i),BGsigma(i),FGmu(i),FGsigma(i),i);
end
figcnt = figcnt+1;
figure(figcnt);
for i = 49:64
    GaussPlot(BGmu(i),BGsigma(i),FGmu(i),FGsigma(i),i);
```

```
end

% pick 8 featrues out of 64 features and plot
% #1,#12,#14,#18,#19,#25,#32,#33
figure('Name','Gaussian Distribution of 8 best features');
best8array = [1,12,14,18,19,25,32,33];
for i = 1:8
    subplot(4,2,i);
    mu1 = BGmu(best8array(i));
    mu2 = FGmu(best8array(i));
    sig1 = BGsigma(best8array(i));
    sig2 = FGsigma(best8array(i));
    xmin = min(mu1,mu2)-2*(sig1+sig2);
    xmax = max(mu1,mu2)+2*(sig1+sig2);
    x = xmin:(xmax-xmin)/100:xmax;
    y1 = (sqrt(2*pi)*sig1).^(-1) * exp(-(x-mu1).^2/(2*sig1*sig1));
    y2 = (sqrt(2*pi)*sig2).^(-1) * exp(-(x-mu2).^2/(2*sig2*sig2));
    plot(x,y1);
    hold on
    plot(x,y2);
    title(best8array(i));
end

% pick 8 featrues out of 64 features and plot
% #2,#3,#4,#5,#59,#60,#63,#64
hold off
figure('Name','Gaussian Distribution of 8 worst features');
worst8array = [2,3,4,5,59,60,63,64];
for i = 1:8
    subplot(4,2,i);
    mu1 = BGmu(worst8array(i));
    mu2 = FGmu(worst8array(i));
    sig1 = BGsigma(worst8array(i));
    sig2 = FGsigma(worst8array(i));
    xmin = min(mu1,mu2)-2*(sig1+sig2);
    xmax = max(mu1,mu2)+2*(sig1+sig2);
    x = xmin:(xmax-xmin)/100:xmax;
    y1 = (sqrt(2*pi)*sig1).^(-1) * exp(-(x-mu1).^2/(2*sig1*sig1));
    y2 = (sqrt(2*pi)*sig2).^(-1) * exp(-(x-mu2).^2/(2*sig2*sig2));
    plot(x,y1);
    hold on
    plot(x,y2);
    title(worst8array(i));
end

% Load the Cheetah.jpg, using DCT and zigzag method
% read the test img cheeta
cheetah_img = imread('cheetah.bmp');
cheetah_dw = im2double(cheetah_img);

% set a blank padding
cheetah_pad = [cheetah_dw, zeros([size(cheetah_dw,1),7]);
zeros([7,size(cheetah_dw,2)+7])];

% initialize test set
img_uzz = zeros([255,270,64]);

% test set implementation
```

```
% slice the image into 8*8 blocks and run the dct2()
for col = (1:size(cheetah_dw,2))
    if (col+7) > size(cheetah_pad,2)
        break;
    end
    for row = (1:size(cheetah_dw,1))
        if (row+7) > size(cheetah_pad,1)
            break;
        end
        temp = reshape(dct2(cheetah_pad(row:row+7,col:col+7)),1,[]);
        img_uzz(row,col,:) = temp;
    end
end

img_uzzR = reshape(img_uzz,[],64);

% load and use zigzag to search the X=? position
zigzag = load('Zig-Zag Pattern.txt');
zigzag = reshape(zigzag, 1, []) + 1;

% the chosen Property in the training set Xtrain -> the chosen Property in
% the testing set Xtest
% (zigzag-ed -> unzigzag-ed)
img_zz = zeros(size(img_uzzR));
for i = 1:size(zigzag,2)
    img_zz(:,zigzag(i)) = img_uzzR(:,i);
end

% Use BDR to decide -64d
img64 = zeros(size(cheetah_dw));
convBG = (BG-BGmu).'*(BG-BGmu)/size(BG,1);
convFG = (FG-FGmu).'*(FG-FGmu)/size(FG,1);

for i = 1:size(img_zz,1)
    bdrBG = ndGauss(img_zz(i,:),BGmu,convBG)*Py_grass;
    bdrFG = ndGauss(img_zz(i,:),FGmu,convFG)*Py_cheetah;
    if (bdrBG/bdrFG)>= 1
        img64(i) = 0;
    else
        img64(i) = 1;
    end
end

img64 = reshape(img64,size(cheetah_dw,1),size(cheetah_dw,2));
img64(size(img64,1)-7:size(img64,1),:) = [];
img64(:,size(img64,2)-7:size(img64,2)) = [];

figure('Name','Cheetah with 64 features');
imagesc(img64);
colormap(gray(255));

% Use BDR to decide -64d
img8 = zeros(size(cheetah_dw));
convBG8 = (BG(:,best8array)-BGmu(:,best8array)).'*(BG(:,best8array)-
BGmu(:,best8array))/size(BG,1);
convFG8 = (FG(:,best8array)-FGmu(:,best8array)).'*(FG(:,best8array)-
FGmu(:,best8array))/size(FG,1);
```

```
for i = 1:size(img_zz,1)
    bdrBG = ndGauss(img_zz(i,best8array),BGmu(best8array),convBG8)*Py_grass;
    bdrFG = ndGauss(img_zz(i,best8array),FGmu(best8array),convFG8)*Py_cheetah;
    if (bdrBG/bdrFG)>= 1
        img8(i) = 0;
    else
        img8(i) = 1;
    end
end

img8 = reshape(img8,size(cheetah_dw,1),size(cheetah_dw,2));
img8(size(img8,1)-7:size(img8,1),:) = [];
img8(:,size(img8,2)-7:size(img8,2)) = [];

figure('Name','Cheetah with 8 best features');
imagesc(img8);
colormap(gray(255));

% accuracy calculate
cheetah_mask = imread('cheetah_mask.bmp');
cheetah_maskdw = im2double(cheetah_mask);
total_num = size(cheetah_maskdw,1)*size(cheetah_maskdw,2);

err_num64 = sum_y(sum_y(abs(cheetah_maskdw(1:247,1:262)-img64),1),2);
err_rate64 = err_num64/total_num;
fprintf('\nError ratio by using 64 properties for Gaussian: %f\n',err_rate64);

err_num8 = sum_y(sum_y(abs(cheetah_maskdw(1:247,1:262)-img8),1),2);
err_rate8 = err_num8/total_num;
fprintf('\nError ratio by using 8 best properties for Gaussian:
%f\n',err_rate8);
```