ECE 271A HW2 Quiz

Chunlin Chen (PID: A59023021)

October 29, 2023

1 Prior Probabilities

1.1 Histogram

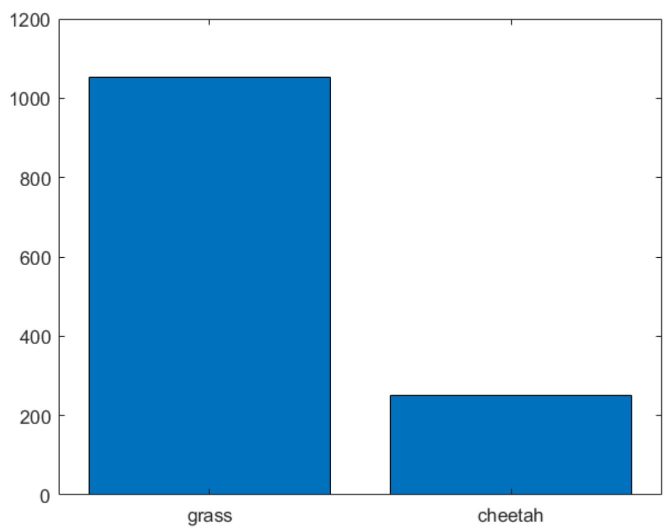


Figure 1: Histogram Estimate of Priors

We first use the training data in `TrainingSamplesDCT_8_new.mat` to compute the histogram estimate of the prior $P_Y(i), i \in \{cheetah, grass\}$, which are just two bar plots as shown in Figure 1.

1.2 MLE for Prior Probabilities

We know that an unbiased estimate for multinomial distribution is that $\hat{P}_X(k) = \frac{c_k}{n}$, where c_k is the number of times that a certain class occurs and n is the number of total observations. Therefore, using the same estimator, we can compute the maximum likelihood estimate of the prior probabilities of the two class:

$$P_Y(cheetah) = \frac{250}{1303} = 0.1919$$

$$P_Y(grass) = \frac{1053}{1303} = 0.8081$$

We can see that what we do above is exactly the same as what we did last week, which is to use the frequency of a certain observed value as the estimate of its probability.

2 Class Conditional Densities

2.1 MLE for 64-Dimensional Gaussian Distribution Parameters

We assume the class-conditional densities follow a 64-dimensional multivariate Gaussian distribution. We know that for a multivariate Gaussian distribution $\boldsymbol{X}$ of $d \times 1$ dimensions, its maximum likelihood estimate given n independent data points is:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{X}^{(i)} = \bar{\boldsymbol{X}}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{X}^{(i)} - \hat{\boldsymbol{\mu}})(\boldsymbol{X}^{(i)} - \hat{\boldsymbol{\mu}})^T$$

here $\hat{\boldsymbol{\mu}}$ is of $d \times 1$ dimensions and $\hat{\boldsymbol{\Sigma}}$ is of $d \times d$ dimensions. Therefore, we can easily compute the MLE for the parameters of the class-conditonal densities $P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ using the training data.

2.2 Feature Marginal Densities

Let's denote the 64×1 vector of DCT coefficients by $\boldsymbol{X} = \{X_1, ..., X_{64}\}$, thereby creating 64 plots of the marginal densities for the two classes.

We will use the following theorem to compute the marginal densities of each feature:

Theorem 1. *Let $\boldsymbol{X}$ follow a multivariate Gaussian distribution:*

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{1}$$

Then, the marginal distribution of of any subset vector $\boldsymbol{X_s}$ is also a multivariate Gaussian distribution:

$$\boldsymbol{X_s} \sim \mathcal{N}(\boldsymbol{\mu_s}, \boldsymbol{\Sigma_s}) \tag{2}$$

where $\boldsymbol{\mu_s}$ drops the irrelevant variables (the ones not in the subset, i.e. marginalized out) from the mean vector $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma_s}$ drops the corresponding rows and columns from the covariance matrix $\boldsymbol{\Sigma}$.

Then, we can plot the marginal densities for different features given each class, as shown in Figure 2-6.

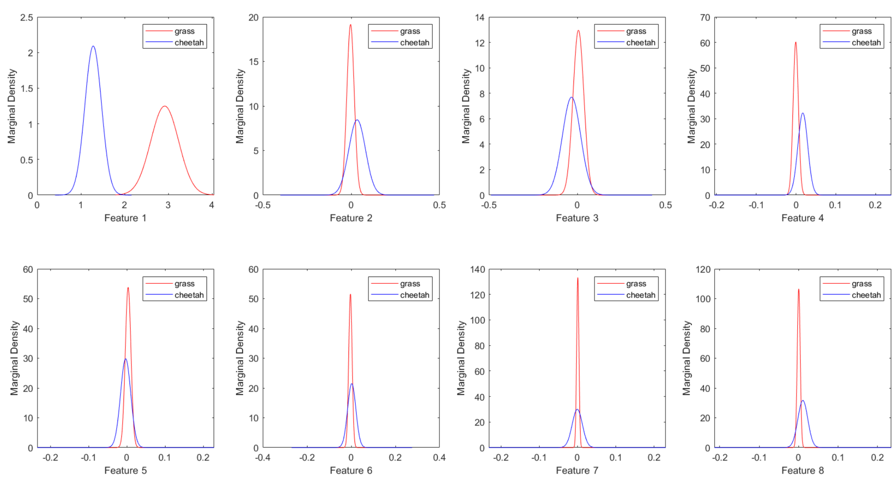


Figure 2: Marginal Densities of Feature 1∼8

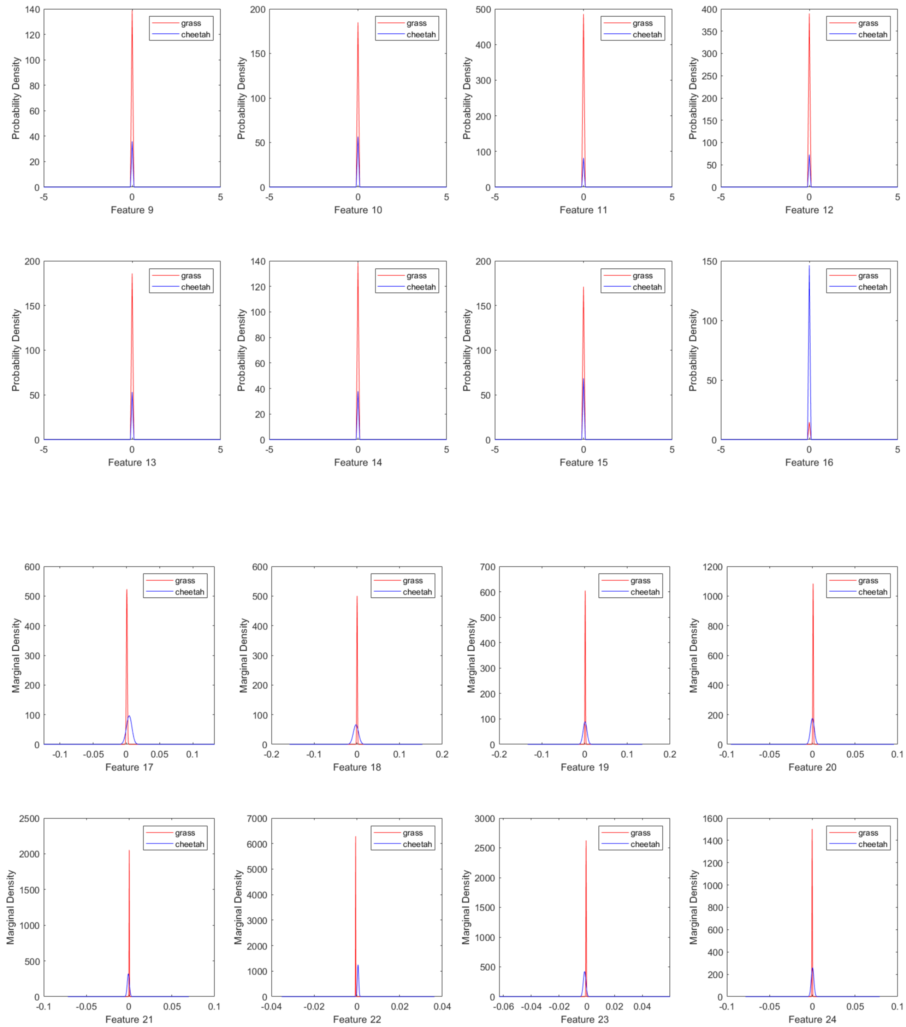


Figure 3: Marginal Densities of Feature 9∼24

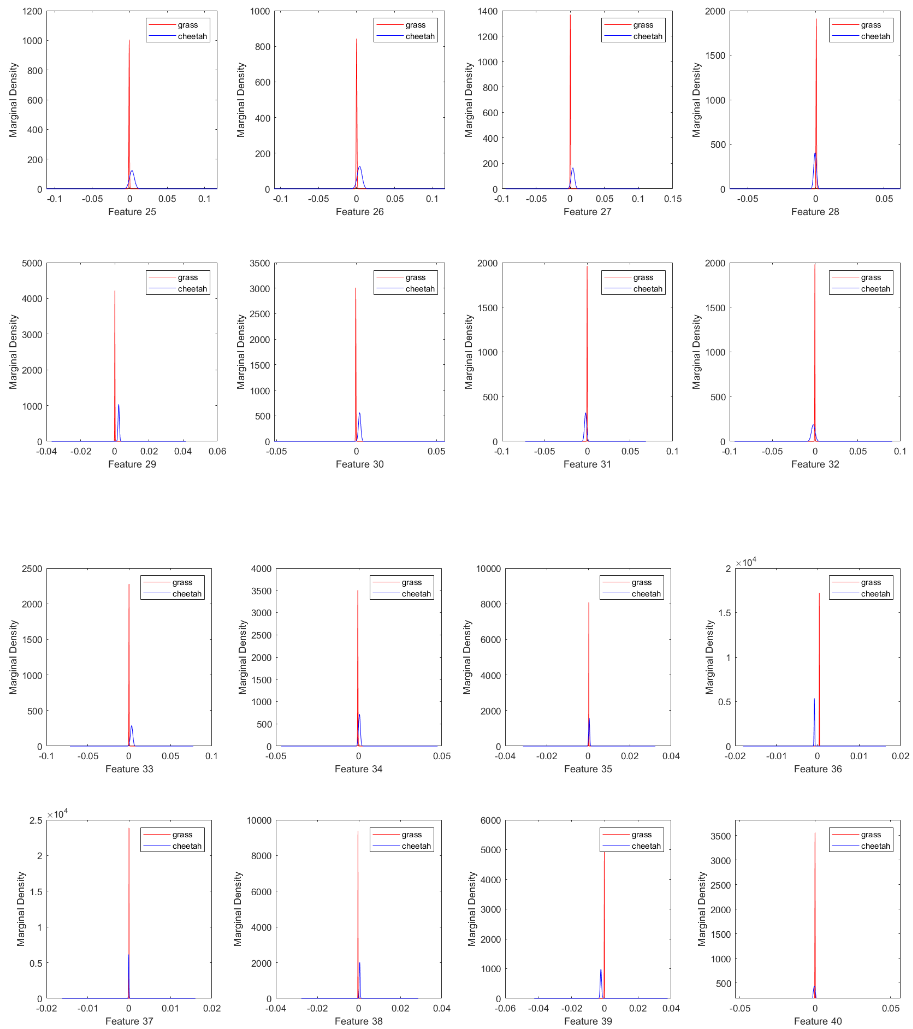


Figure 4: Marginal Densities of Feature 25∼40

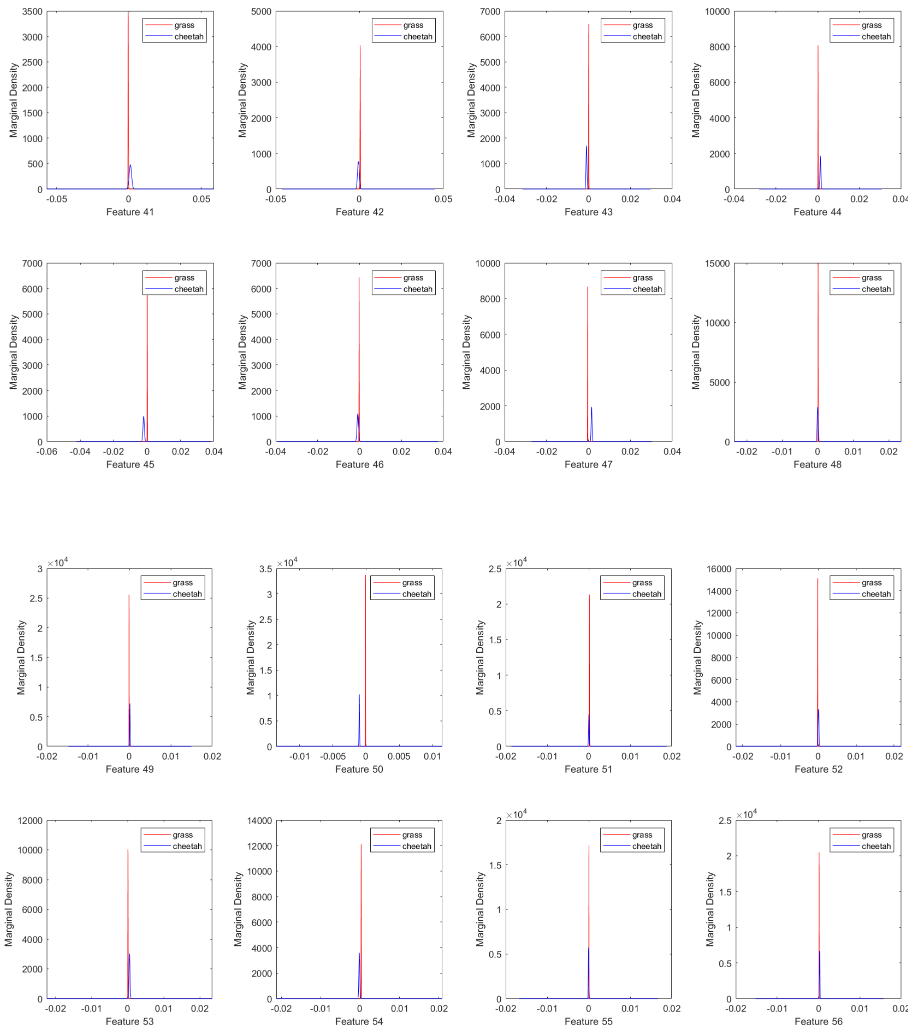


Figure 5: Marginal Densities of Feature 41∼56

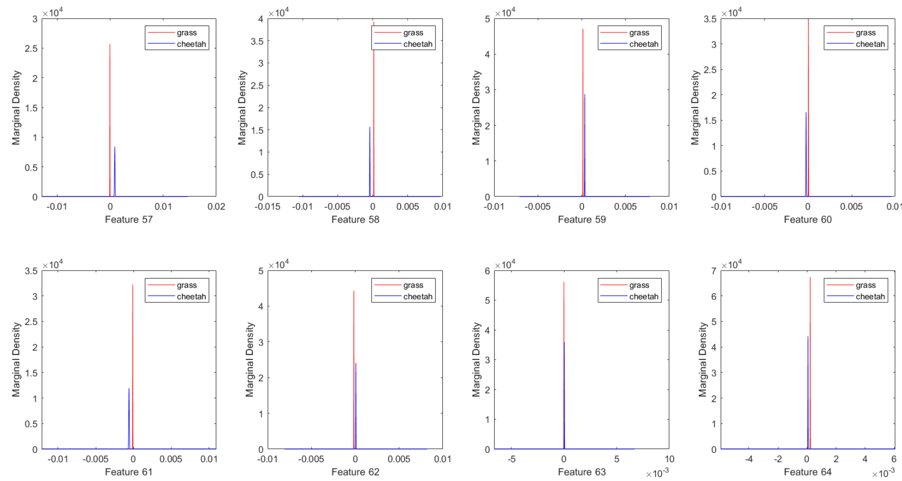


Figure 6: Marginal Densities of Feature 57∼64

By to visual inspection, we select the best 8 features and the worst 8 features, based on the criterion that whether the marginal densities given the two classes are separated distinctly. The results are shown in Figure 7-8.

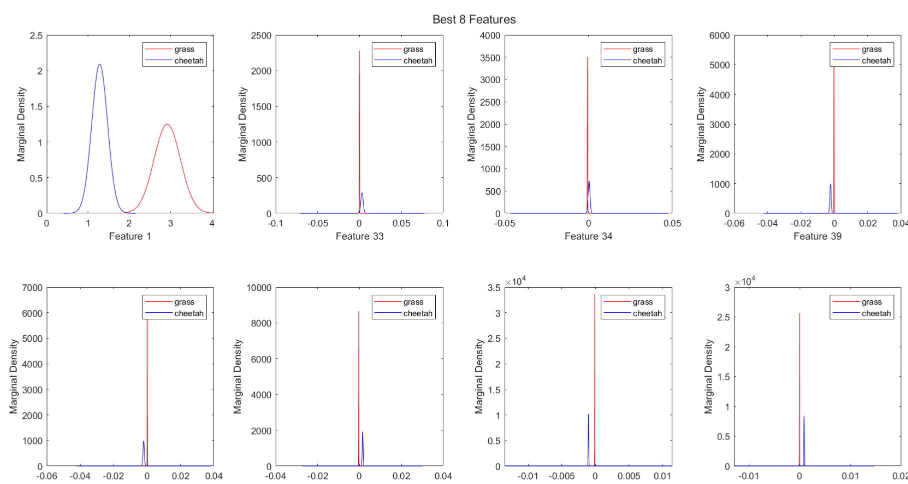


Figure 7: Best 8 Features

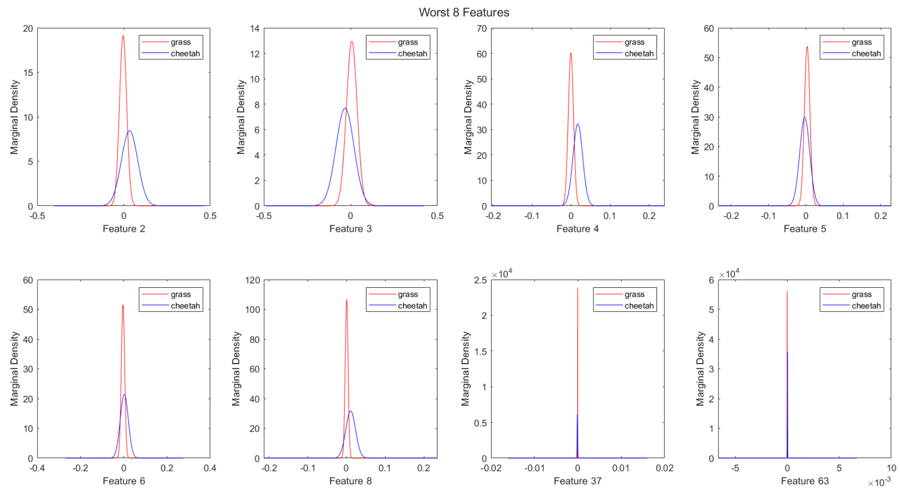


Figure 8: Worst 8 Features

3 Classification

Using the Bayesian Decision Rule for multivariate Gaussian, which is

$$i^*(\boldsymbol{x}) = \underset{i}{argmax}[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_i})^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu_i}) - \frac{1}{2}\log(2\pi)^d|\boldsymbol{\Sigma}_i| + \log P_Y(i)]$$

, we can create classification masks for the test image.

3.1 64-Dimensional Gaussian

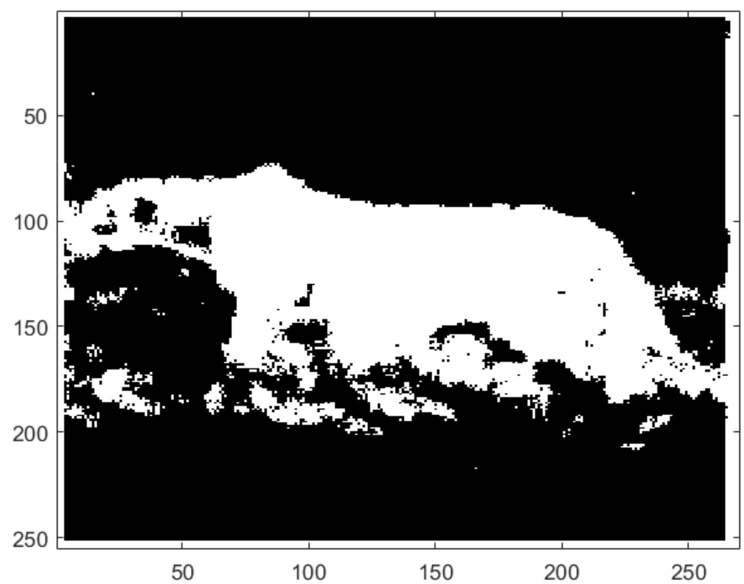
We first use the 64-dimensional Gaussian, and the classification result is shown in Figure 9. The detection rate is $P_{X|Y}(g(x) = cheetah|cheetah) = 0.9444$ and the false alarm rate is $P_{X|Y}(g(x) = cheetah|grass) = 0.1549$.

Then we can calculate the probability of error:

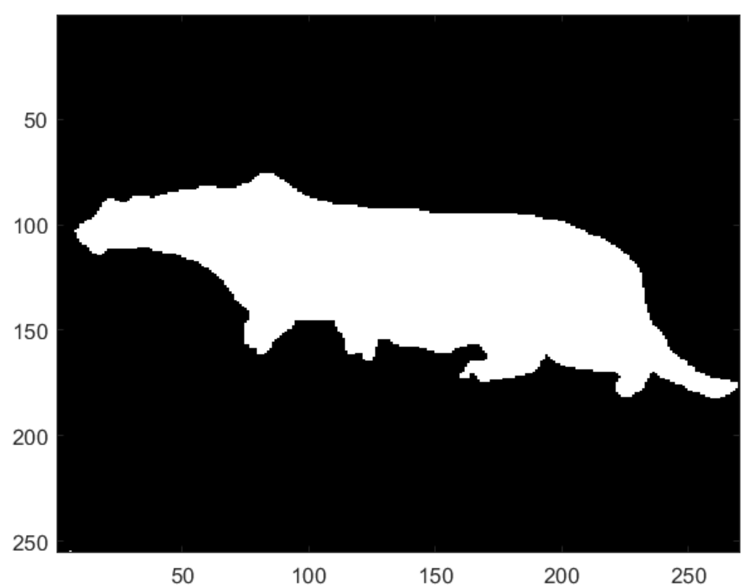
$$P_E = E_Y[P_{X|Y}(g(x) \neq Y|Y)] = \sum_i P_{X|Y}(g(x) \neq i|i)P_Y(i)$$

$$= P_{X|Y}(g(x) = cheetah|grass)P_Y(grass) + P_{X|Y}(g(x) = grass|cheetah)P_Y(cheetah)$$

$$= 0.1549 * 0.8081 + (1 - 0.9444) * 0.1919 = 0.1358$$

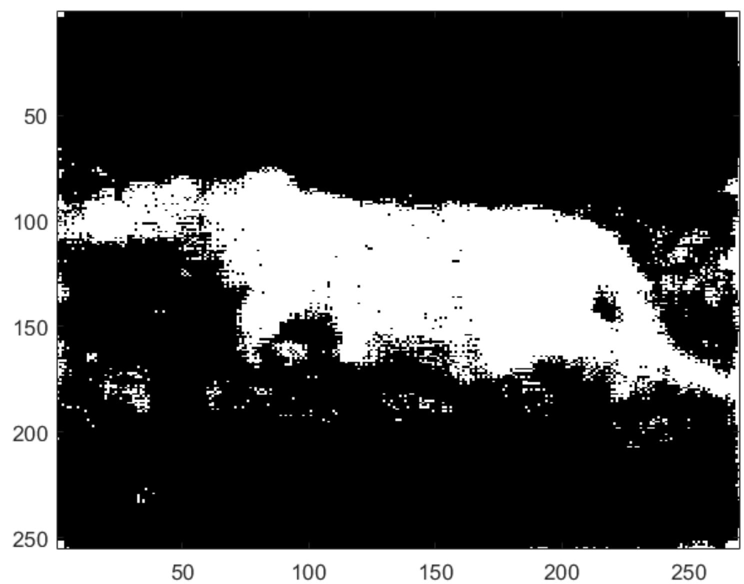


(a) Classification Mask

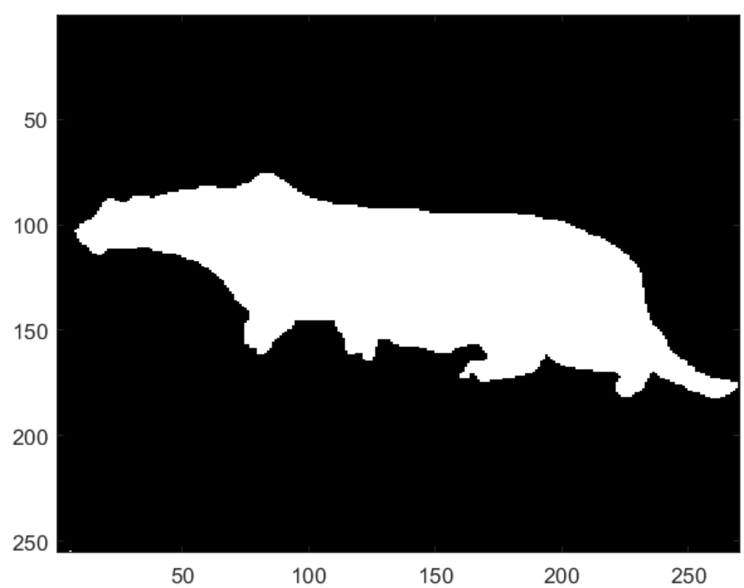


(b) Ground Truth

Figure 9: Classification Result using 64D Gaussian



(a) Classification Mask



(b) Ground Truth

Figure 10: Classification Result using 8D Gaussian

3.2 8-Dimensional Gaussian

Then we only use the best 8 features for classification, and the result is shown in Figure 10. Obviously in this way the classification mask has much less noise than using 64D Gaussian.

And the detection rate is $P_{X|Y}(g(x) = cheetah|cheetah) = 0.9272$ and the false alarm rate is $P_{X|Y}(g(x) = cheetah|grass) = 0.0447$. The probability of error is $P_E = 0.0447 * 0.8081 + (1 - 0.9272) * 0.1919 = 0.0501$.

We can see that using the best 8 features for classification can significantly lower the false alarm rate since in this way the classifier can rule out the features that fail to exhibit distinct distribution under the two classes and thus lower the error rate. However, the detection rate using all 64 features is slightly higher than that of the 8D Gaussian, which is because using only the 8 best features of a pixel block may lose some information, leading to the classifier having a slightly lower detection rate. Even though, in term of the overall probability of error, the 8D Gaussian significantly outperforms the 64D Gaussian, together with a much lower computational complexity (only one eighth of the features are used for classification), indicating that using distinctive features, instead of all features without excluding the reluctant ones, can achieve much better classification performance.

4 Source Code

```
load("HW2\TrainingSamplesDCT_8_new.mat")

% Compute priors using MLE results in Prob 2
c_fg = size(TrainsampleDCT_FG, 1);
c_bg = size(TrainsampleDCT_BG, 1);
n = c_fg + c_bg;
X = categorical({'grass', 'cheetah'});
X = reordercats(X, {'grass', 'cheetah'});
Y = [c_bg c_fg];
bar(X, Y)
prob_bg = c_bg / n
prob_fg = c_fg / n

% MLE for class conditional densities
```

```
% BG
mean_bg_est = mean(TrainsampleDCT_BG, 1)
var_bg_est = 0;
for i = 1:size(TrainsampleDCT_BG, 1)
    var_bg_est = var_bg_est + (TrainsampleDCT_BG(i, :) - mean_bg_est).'...
        *(TrainsampleDCT_BG(i, :) - mean_bg_est)./size(TrainsampleDCT_BG, 1);
end
var_bg_est
% FG
mean_fg_est = mean(TrainsampleDCT_FG, 1)
var_fg_est = 0;
for i = 1:size(TrainsampleDCT_FG, 1)
    var_fg_est = var_fg_est + (TrainsampleDCT_FG(i, :) - mean_fg_est).'...
        *(TrainsampleDCT_FG(i, :) - mean_fg_est)./size(TrainsampleDCT_FG, 1);
end
var_fg_est

% Plot marginal densities for all features
for j = 1:8
    figure(j);
    set(gcf,'outerposition',get(0,'screensize'));
    tiledlayout(2,4)
    for i = (j-1)*8+1:j*8
        nexttile
        x_low_bg = mean_bg_est(1, i) - 2 * sqrt(var_bg_est(i, i));
        x_low_fg = mean_fg_est(1, i) - 2 * sqrt(var_fg_est(i, i));
        x_up_bg = mean_bg_est(1, i) + 2 * sqrt(var_bg_est(i, i));
        x_up_fg = mean_fg_est(1, i) + 2 * sqrt(var_fg_est(i, i));
        p_bg = normpdf(x_low_bg:1e-6:x_up_bg, mean_bg_est(1, i), var_bg_est(i, i));
```

```
        p_fg = normpdf(x_low_fg:1e-6:x_up_fg,
           mean_fg_est(1, i), var_fg_est(i, i));
        plot(x_low_bg:1e-6:x_up_bg, p_bg, 'r');
        hold on
        plot(x_low_fg:1e-6:x_up_fg, p_fg, 'b');
        xlabel(append('Feature ', int2str(i)))
        ylabel('Marginal Density')
        legend('grass','cheetah')
    end
end

% best 8 features
best_8_indices = [1 33 34 39 45 47 50 57];
figure(9)
set(gcf,'outerposition',get(0,'screensize'));
tiledlayout(2,4)
for i = 1:size(best_8_indices, 2)
    nexttile
    x_low_bg = mean_bg_est(1, best_8_indices(1, i))
       - 2 * sqrt(var_bg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
    x_low_fg = mean_fg_est(1, best_8_indices(1, i))
       - 2 * sqrt(var_fg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
    x_up_bg = mean_bg_est(1, best_8_indices(1, i)) +
        2 * sqrt(var_bg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
    x_up_fg = mean_fg_est(1, best_8_indices(1, i)) +
        2 * sqrt(var_fg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
    p_bg = normpdf(x_low_bg:1e-6:x_up_bg,
       mean_bg_est(1, best_8_indices(1, i)),
       var_bg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
    p_fg = normpdf(x_low_fg:1e-6:x_up_fg,
       mean_fg_est(1, best_8_indices(1, i)),
       var_fg_est(best_8_indices(1, i),
       best_8_indices(1, i)));
```

```
    plot(x_low_bg:1e-6:x_up_bg, p_bg, 'r');
    hold on
    plot(x_low_fg:1e-6:x_up_fg, p_fg, 'b');
    xlabel(append('Feature ', int2str(best_8_indices(1, i))))
    ylabel('Marginal Density')
    legend('grass','cheetah')
end
sgtitle('Best 8 Features')

% worst 8 features
worst_8_indices = [2 3 4 5 6 8 37 63];
figure(10)
set(gcf,'outerposition',get(0,'screensize'));
tiledlayout(2,4)
for i = 1:size(worst_8_indices, 2)
    nexttile
    x_low_bg = mean_bg_est(1, worst_8_indices(1, i)) - 2 * sqrt(var_bg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    x_low_fg = mean_fg_est(1, worst_8_indices(1, i)) - 2 * sqrt(var_fg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    x_up_bg = mean_bg_est(1, worst_8_indices(1, i)) + 2 * sqrt(var_bg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    x_up_fg = mean_fg_est(1, worst_8_indices(1, i)) + 2 * sqrt(var_fg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    p_bg = normpdf(x_low_bg:1e-6:x_up_bg, mean_bg_est(1, worst_8_indices(1, i)), var_bg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    p_fg = normpdf(x_low_fg:1e-6:x_up_fg, mean_fg_est(1, worst_8_indices(1, i)), var_fg_est(worst_8_indices(1, i), worst_8_indices(1, i)));
    plot(x_low_bg:1e-6:x_up_bg, p_bg, 'r');
```

```
    hold on
    plot(x_low_fg:1e-6:x_up_fg, p_fg, 'b');
    xlabel(append('Feature ', int2str(worst_8_indices(1, i))))
    ylabel('Marginal Density')
    legend('grass','cheetah')
end
sgtitle('Worst 8 Features')

ZigZagPattern = readmatrix('HW1\Zig-Zag Pattern.txt');
ZigZagPattern = ZigZagPattern + 1;
ZigZagPattern = int8(ZigZagPattern);

img = imread("HW1\cheetah.bmp");
img = im2double(img);

% Zero Padding
% right = zeros(255, 7);
% bottom = zeros(7, 277);
% img_pad = [[img right]; bottom];
left = zeros(255, 3);
right = zeros(255, 4);
up = zeros(3, 277);
bottom = zeros(4, 277);
img_pad = [up; [left img right]; bottom];

% DCT
img_dct = dct_8(img, img_pad);

% ZigZag Scan
img_scan = blockproc(img_dct, [8 8], @(block_struct) ZigZagScan(block_struct.data, ZigZagPattern));

% BDR 64D Gaussian
mask_64 = blockproc(img_scan, [1, 64], @(block_struct) BDR(block_struct.data, mean_bg_est, mean_fg_est, var_bg_est, var_fg_est, prob_bg,
```

```
    prob_fg));
mask_64 = int8(mask_64);

% BDR 8D Gaussian
mean_bg_est_8 = best_8_v(mean_bg_est, best_8_indices
    );
mean_fg_est_8 = best_8_v(mean_fg_est, best_8_indices
    );
var_bg_est_8 = best_8_m(var_bg_est, best_8_indices);
var_fg_est_8 = best_8_m(var_fg_est, best_8_indices);
img_scan_8 = blockproc(img_scan, [1 64], @(
    block_struct) best_8_v(block_struct.data,
    best_8_indices));
mask_8 = blockproc(img_scan_8, [1, 8], @(
    block_struct) BDR(block_struct.data,
    mean_bg_est_8, mean_fg_est_8, var_bg_est_8,
    var_fg_est_8, prob_bg, prob_fg));
mask_8 = int8(mask_8);

ground_truth = imread("HW1\cheetah_mask.bmp");
ground_truth = im2double(ground_truth);
close all
imagesc(mask_64)
colormap(gray(255))
imagesc(mask_8)
colormap(gray(255))
imagesc(ground_truth)
colormap(gray(255))
ground_truth = int8(ground_truth);

diff_64 = ground_truth - mask_64;
diff_8 = ground_truth - mask_8;
detect_64 = 1 - sum(sum(diff_64==1))/sum(sum(
    ground_truth==1))
detect_8 = 1 - sum(sum(diff_8==1))/sum(sum(
    ground_truth==1))
Falarm_64 = sum(sum(diff_64==-1))/sum(sum(
    ground_truth==0))
```

```
Falarm_8 = sum(sum(diff_8==-1))/sum(sum(ground_truth==0))
p_error_64 = Falarm_64 * prob_bg + (1 - detect_64) * prob_fg
p_error_8 = Falarm_8 * prob_bg + (1 - detect_8) * prob_fg

function vector = ZigZagScan(matrix, pattern)
    vector = zeros(1, size(matrix, 1) * size(matrix, 2));
    for i = 1:size(matrix, 1)
        for j = 1:size(matrix, 2)
            position = pattern(i, j);
            vector(1, position) = matrix(i, j);
        end
    end
end

function mask = BDR(feature, mu_bg, mu_fg, sigma_bg, sigma_fg, P_bg, P_fg)
    if (feature-mu_bg)*inv(sigma_bg)*(feature-mu_bg).'+log((2*pi).^64*det(sigma_bg))-2*log(P_bg) ...
            < (feature-mu_fg)*inv(sigma_fg)*(feature-mu_fg).'+log((2*pi).^64*det(sigma_fg))-2*log(P_fg)
        mask = 0;
    else
        mask = 1;
    end
end

function dct = dct_8(img, img_pad)
    dct = zeros(size(img, 1) * 8, size(img, 2) * 8);
    for i = 1:size(img, 1)
        for j = 1:size(img, 2)
            dct((8*i-7):(8*i), (8*j-7):(8*j)) = dct2(img_pad(i:i+7, j:j+7));
```

```
        end
    end
end

function vector = best_8_v(feat_64, indices)
    vector = feat_64;
    unwanted = setdiff([1:64], indices);
    vector(:, unwanted) = [];
end

function matrix = best_8_m(feat_64, indices)
    matrix = feat_64;
    unwanted = setdiff([1:64], indices);
    matrix(:, unwanted) = [];
    matrix(unwanted, :) = [];
end
```