Assignment - 2

Srinidhi Bharadwaj Kalgundi Srinivas

A59010584

Type *Markdown* and LaTeX: α^2

a) Using the training data in TrainingSamplesDCT 8.mat compute the histogram estimate of the prior P_Y (i), where i belongs to {cheetah, grass}

Solution: Prior probabilities are calculate using maximum likelihood estimates from problem 2 of HW are;

Calculated prior probabilites are:

P_Y (Cheetah) is 0.1919

P_Y (Grass) is 0.8081

Result obtained is the same as that of quiz 1. Calculation of prior was done in an intuitive way for quiz 1 where as in this problem the calculation of prior is backed my math, using MLE.

Matlab code snippet:

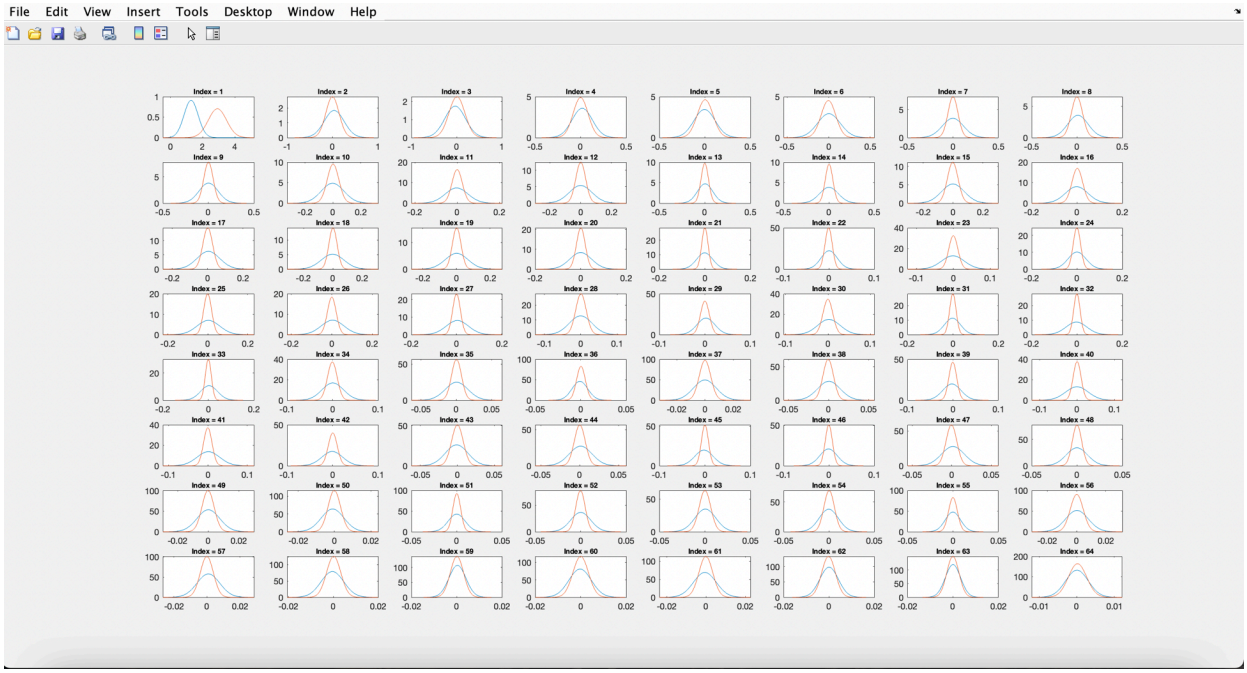
```
% Load training set

TrainingSet = load("TrainingSamplesDCT_8_new.mat");
foreground = TrainingSet.TrainsampleDCT_FG;
background = TrainingSet.TrainsampleDCT_BG;

% ------------------------------------------------------------------- %
% Problem a

[fg_rows, fg_cols] = size(foreground);
[bg_rows, bg_cols] = size(background);

total_samples = fg_rows + bg_rows;

prior_foreground = fg_rows / total_samples;
prior_background = bg_rows / total_samples;
```

Type *Markdown* and LaTeX: α^2

b) Using the training data in TrainingSamplesDCT 8.mat, compute the maximum likelihood estimates for the parameters of the class conditional densities $P_{X|Y}$ (x|cheetah) and $P_{X|Y}$ (x|grass) under the

Gaussian assumption.

Solution: Maximum likelihood estimates for the class conditional densities under Gaussian assumption are calculated. Parameters calculated are (mean, covariance). Below is the image for marginal densities for $P_{X_k|Y}$(x|cheetah) and $P_{X_k|Y}$(x|grass) where k = {1,2,..,64}

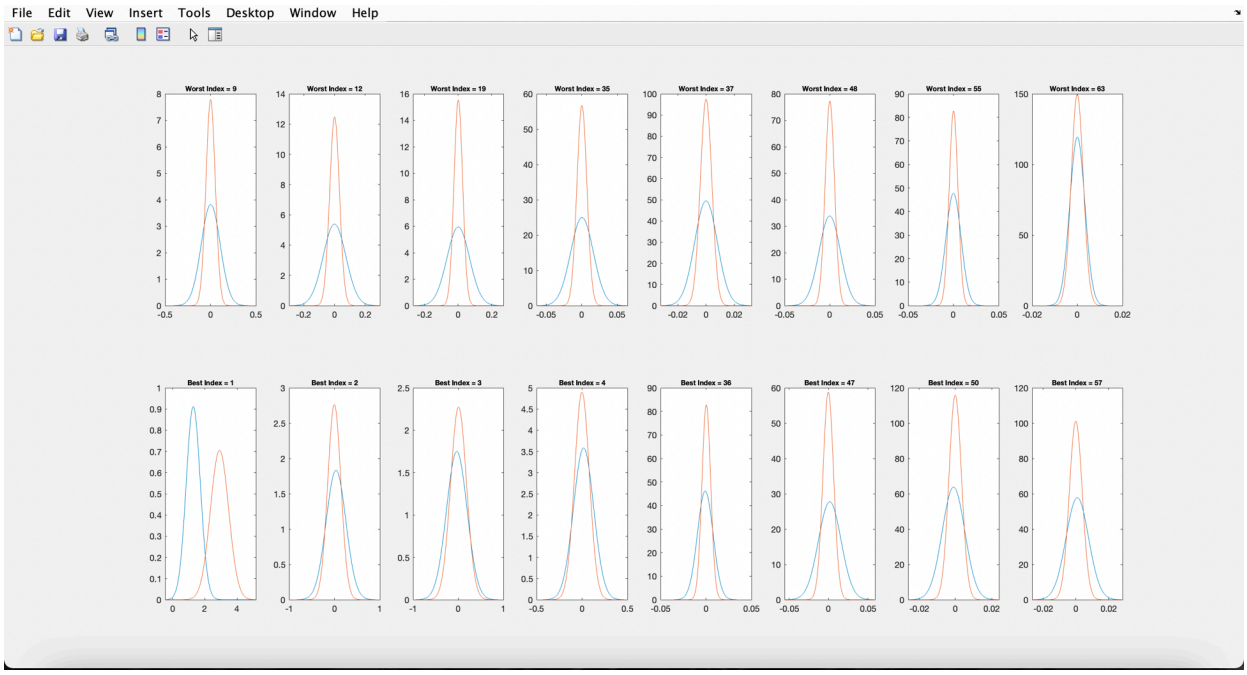


The best 8 features and the worst 8 features for classification are calculated using the z-test comparision of 2 probability density functions and the following indices are selected.

Indices of the best 8 features = {1, 2, 3, 4, 36, 47, 50, 57}

Indices of the worst 8 features = {9, 12, 19, 35, 37, 48, 55, 63}

Below is the image for marginal densities for $P_{X_k|Y}$(x|cheetah) and $P_{X_k|Y}$(x|grass) for 8 best and worst features.



Type *Markdown* and LaTeX: α^2

c) Compute the Bayesian decision rule and classify the locations of the cheetah image using

```
i) the 64-dimensional Gaussians
ii) the 8-dimensional Gaussians associated with the best 8 fe
atures.
```

For the two cases, plot the classi cation masks and compute the probability of error by comparing with cheetah mask.bmp. Can you explain the results?

Solution: i) the 64-dimensional Gaussians

- Read the given image and convert it to double (DCT output may be type casted to int otherwise)
- Pad the images with 7 zeros on all 4 sides to maintain the original image size after calculation and to ensure the sliding window stays within bounds
- Pick an 8x8 block from the cheetah image and calculate the DCT of the image
- Convert the output(8x8) into zigzag format using the provided zigzag pattern file
- Use the computed zigzag format as the feature vector X for the 64 dimensional Gaussian PDF. Mean and covariance of this PDF are calulated using the maximum likelihood estimates
- Use the n-dimensional Gaussian distrubution function;

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

where,

- d = 64
- i = class
- μ = mean
- $\sum$ = covariance
- x = feature vector (obtained from DCT)

- After estimating the class conditional density, it is multiplied with the prior probability to get the posterior value of a given feature being cheetah or grass.
- Both $P_{(Y|X)}(cheetah|x)$ and $P_{(Y|X)}(grass|x)$ are calculated and the decision is chosen based on Bayes rule, i.e., a pixel in "result_64" vector is set if $P_{(Y|X)}(cheetah|x) > P_{(Y|X)}(grass|x)$ and the mask is obtained.

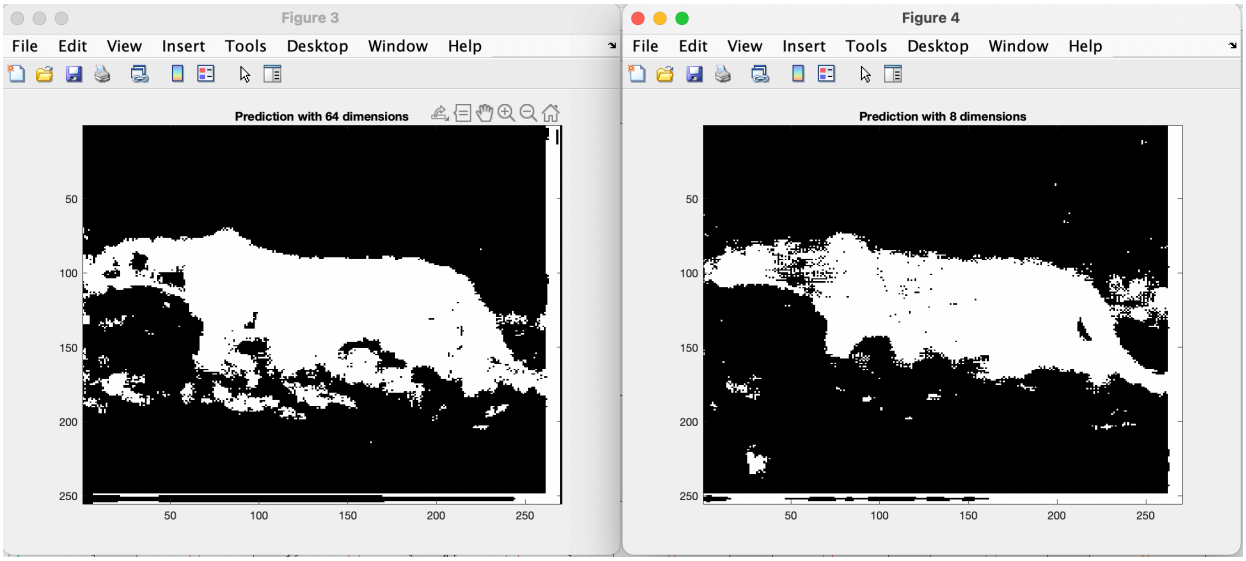
ii) the 8-dimensional Gaussians associated with the best 8 features.

- Best 8 dimensional gaussians are calculated using the z-test and used for calculated the masks
- After calculating the DCT, best 8 features are selected using the best index vector calculated in the previous problem
- Mean and covariances are also selected using the best index vector
- 8-dimensional Gaussian is calculate and multipled with the prior probabilities for both background and foreground using their respective mean and covariances.

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

where,

- d = 8
- i = class
- μ = mean
- $\sum$ = covariance
- x = feature vector (obtained from DCT)



Error calculation:

- Image the ground truth image into the workspace
- Loop through all the pixels of the ground truth image and count the number of foreground and backgrond images. This will be used for percentage error calculation
- Loop through the pixels of the array A (predicted_image in the code) and compare each pixel of array A with the ground truth and calculate the number of mis-classified pixels
- Calculate foreground error as (mis-classified foreground/total foreground pixels) * prior(foreground)
- Calculate background error as (mis-classified background/total background pixels) * prior(background)
- Calculate total error as the sum of results of above 2 steps

Computed errors:

- Probability of error for mask obtained using 64 dimensional Gaussian is: **12.2396%**
- Probability of error for mask obtained using 8 dimensional Gaussian is: **10.1583%**

Comparing the two images, it is clear that working with the best 8 features provide slightly improved results, especially near the legs of the cheetah where background pixels are mis-classified as foreground. When entire 64 features are used, the worst features will result in the data not contributing much to the posterior probability calculation and the decision becomes more biased towards the prior probabilities. Since the prior probability of background (grass) is greater than that of the foreground(cheetah), there are more background pixels than foreground pixels. Same is observed in the error.

Note: $P_{(Y|X)}(cheetah|x) = P_{X|Y}$(x|cheetah) * P_Y(cheetah) and $P_{(Y|X)}(grass|x) = P_{X|Y}$(x|grass) * P_Y(grass)

Type *Markdown* and LaTeX: α^2

Matlab code:

```
clc
clear
close all

% Load training set

TrainingSet = load("TrainingSamplesDCT_8_new.mat");
foreground = TrainingSet.TrainsampleDCT_FG;
background = TrainingSet.TrainsampleDCT_BG;

% ------------------------------------------------------------
---------- %
% Problem a

[fg_rows, fg_cols] = size(foreground);
[bg_rows, bg_cols] = size(background);

total_samples = fg_rows + bg_rows;

prior_foreground = fg_rows / total_samples;
prior_background = bg_rows / total_samples;


% ------------------------------------------------------------
---------- %
% Problem b

% Foreground and background means %
mu_foreground = mean(foreground);
mu_background = mean(background);

% Foreground and background variances %
var_foreground = var(foreground);
var_background = var(background);

% Foreground and background covariances %
covar_foreground = cov(foreground);
```

```
covar_background = cov(background);
wi_foreground = inv(covar_foreground);
wi_background = inv(covar_background);

z_distance = [];
for idx=1:fg_cols
    subplot(8,8,idx)
    set(gcf,'Position', get(0, 'ScreenSize'))

    fg_average = mean(foreground(:,idx));
    fg_variance = var(foreground(:,idx));
    fg_standard_deviation = sqrt(fg_variance);

    bg_average = mean(background(:,idx));
    bg_variance = var(background(:,idx));
    bg_standard_deviation = sqrt(bg_variance);

    % Considering 4 standard deviations for the Gaussian
    fore = fg_average-4*fg_standard_deviation:fg_standard_devi
ation/50:...
                        fg_average+4*fg_standard_deviation;
    back = bg_average-4*bg_standard_deviation:bg_standard_devi
ation/50:...
                        bg_average+4*bg_standard_deviation;
    x = sort([fore back]); % x-axis
    p_x_y_cheetah = gaussian(x,fg_average,fg_standard_deviatio
n);
    p_x_y_grass = gaussian(x,bg_average,bg_standard_deviation)
;

    plot(x,p_x_y_cheetah,x,p_x_y_grass)
    title(['Index = ' num2str(idx)],'FontSize',8);

    % z-test for comparision
    z_tmp = z_test(fg_variance, fg_average, bg_variance, bg_av
erage);
    z_distance = [z_distance z_tmp];
end

[z_sorted, z_idx] = sort(z_distance);
worstidx = sort(z_idx(1:8)); % Worst Index values
bestidx = sort(z_idx(57:64)); % Best Index values
```

```
figure();
for i=1:8
    subplot(2,8,i)
    worst_idx = worstidx(i);
    set(gcf,'Position', get(0, 'ScreenSize'))
    fg_average = mean(foreground(:,worst_idx));
    fg_variance = var(foreground(:,worst_idx));
    fg_standard_deviation = sqrt(fg_variance);

    bg_average = mean(background(:,worst_idx));
    bg_variance = var(background(:,worst_idx));
    bg_standard_deviation = sqrt(bg_variance);
    % Considering 4 standard deviations for the Gaussian
    fore = fg_average-4*fg_standard_deviation:fg_standard_devi
ation/50:...
                        fg_average+4*fg_standard_deviation;
    back = bg_average-4*bg_standard_deviation:bg_standard_devi
ation/50:...
                        bg_average+4*bg_standard_deviation;
    x = sort([fore back]); % x-axis
    p_x_y_cheetah = gaussian(x,fg_average,fg_standard_deviatio
n);
    p_x_y_grass = gaussian(x,bg_average,bg_standard_deviation)
;

    plot(x,p_x_y_cheetah,x,p_x_y_grass)
    title(['Worst Index = ' num2str(worst_idx)],'FontSize',8);
end

for i=1:8
    subplot(2,8,i+8)
    %w = get(0, 'ScreenSize');
    best_idx = bestidx(i);
    set(gcf,'Position', get(0, 'ScreenSize'))
    fg_average = mean(foreground(:,best_idx));
    fg_variance = var(foreground(:,best_idx));
    fg_standard_deviation = sqrt(fg_variance);

    bg_average = mean(background(:,best_idx));
    bg_variance = var(background(:,best_idx));
    bg_standard_deviation = sqrt(bg_variance);
```

```
    % Considering 4 standard deviations for the Gaussian
    fore = fg_average-4*fg_standard_deviation:fg_standard_devi
ation/50:...
                        fg_average+4*fg_standard_deviation;
    back = bg_average-4*bg_standard_deviation:bg_standard_devi
ation/50:...
                        bg_average+4*bg_standard_deviation;
    x = sort([fore back]); % x-axis
    p_x_y_cheetah = gaussian(x,fg_average,fg_standard_deviatio
n);
    p_x_y_grass = gaussian(x,bg_average,bg_standard_deviation)
;

    plot(x,p_x_y_cheetah,x,p_x_y_grass)
    title(['Best Index = ' num2str(best_idx)],'FontSize',8);
end

% ------------------------------------------------------------
---------- %
% Problem c

pad_value = 7;
cheetah = imread("../cheetah.bmp");
cheetah = padarray(cheetah,[pad_value pad_value], 'post');
cheetah = im2double(cheetah);

[rows, cols] = size(cheetah);

unpadded_rows = rows-pad_value;
unpadded_cols = cols - pad_value;
converted_block = zeros(unpadded_rows, unpadded_cols);
zigzag = load("../Zig-Zag Pattern.txt");
zigzag = zigzag + 1; %Following MATLAB index
result_64 = zeros(unpadded_rows, unpadded_cols);
mu_foreground = transpose(mu_foreground);
mu_background = transpose(mu_background);
determinant_foreground = det(covar_foreground);
determinant_background = det(covar_background);

% Prediction with 64 dimensional gaussians
for i = 1:unpadded_rows
    for j = 1:unpadded_cols
```

```
        block = cheetah(i:i+7, j:j+7);
        transform = dct2(block);
        % Create zigzag pattern
        zigzag_transform(zigzag) = transform;
        x = transpose(zigzag_transform);

        % Cheetah
        exp_func = -0.5 * transpose(x-mu_foreground) * wi_fore
ground * (x-mu_foreground);
        exp_coeff = 1/(sqrt(((2*pi)^64) * determinant_foregrou
nd));
        p_y_x_cheetah = exp_coeff * exp(exp_func) * prior_fore
ground;

        % Grass
        exp_func = -0.5 * transpose(x-mu_background) * wi_back
ground * (x-mu_background);
        exp_coeff = 1/(sqrt(((2*pi)^64) * determinant_backgrou
nd));
        p_y_x_grass = exp_coeff * exp(exp_func) * prior_backgr
ound;

        if(p_y_x_cheetah > p_y_x_grass)
            result_64(i, j) = 1;
        end
    end
end

result_8 = zeros(unpadded_rows, unpadded_cols);
mu_foreground_best = transpose(mean(foreground(:, bestidx)));
mu_background_best = transpose(mean(background(:, bestidx)));

cov_foreground_best = cov(foreground(:, bestidx));
cov_background_best = cov(background(:, bestidx));

wi_foreground_best = inv(cov_foreground_best);
wi_background_best = inv(cov_background_best);

determinant_best_fg = det(cov_foreground_best);
determinant_best_bg = det(cov_background_best);

% Prediction with 8 dimensional gaussians
```

```
for i = 1:unpadded_rows
    for j = 1:unpadded_cols
        block = cheetah(i:i+7, j:j+7);
        transform = dct2(block); % Utilize inbuilt MATLAB func
tion for DCT calculation
        % Create zigzag pattern
        zigzag_transform(zigzag) = transform;
        x = transpose(zigzag_transform(:, bestidx));

        % Cheetah
        exp_func = -0.5 * transpose(x-mu_foreground_best) * wi
_foreground_best ...
                                                * (x-mu_fore
ground_best);
        exp_coeff = 1/(sqrt(((2*pi)^8) * determinant_best_fg))
;
        p_y_x_cheetah = exp_coeff * exp(exp_func) * prior_fore
ground;

        % Grass
        exp_func = -0.5 * transpose(x-mu_background_best) * wi
_background_best ...
                                                * (x-mu_back
ground_best);
        exp_coeff = 1/(sqrt(((2*pi)^8) * determinant_best_bg))
;
        p_y_x_grass = exp_coeff * exp(exp_func) * prior_backgr
ound;
        if(p_y_x_cheetah > p_y_x_grass)
            result_8(i, j) = 1;
        end
    end
end

figure;
imagesc(result_64);
title('Prediction with 64 dimensions')
colormap(gray(255));

figure;
imagesc(result_8);
title('Prediction with 8 dimensions')
```

```
colormap(gray(255));

% Error calculation

true_image = imread('../cheetah_mask.bmp');
error_64_dim = calc_error(true_image, result_64, prior_foregro
und, prior_background);
X = ['Probability of error for mask obtained using 64 dimensio
nal Gaussian ' ...
           'is: ', num2str(error_64_dim), '%'];
disp(X)

error_8_dim = calc_error(true_image, result_8, prior_foregroun
d, prior_background);
X = ['Probability of error for mask obtained using 8 dimension
al Gaussian ' ...
           'is: ', num2str(error_8_dim), '%'];
disp(X)

% ----------------------------------------------------------
---------- %
% Function to calculate the error
function error = calc_error(ref_image, actual_image, p_fore, p
_back)
    foreground_pixels = 0;
    background_pixels = 0;

    for i=1:size(ref_image, 1)
        for j=1:size(ref_image, 2)
            if ref_image(i, j) == 255
                foreground_pixels = foreground_pixels + 1;
            else
                background_pixels = background_pixels + 1;
            end
        end
    end

    foreground_error = 0;
    background_error = 0;
    for i=1:size(actual_image, 1)
        for j=1:size(actual_image, 2)
            if ref_image(i, j) == 255 && actual_image(i, j) ==
```

```
0
                foreground_error = foreground_error + 1;
            elseif ref_image(i, j) == 0 && actual_image(i, j)
== 1
                background_error = background_error + 1;
            end
        end
    end

    error_foreground = (foreground_error / foreground_pixels)
* p_fore;
    error_background = (background_error / background_pixels)
* p_back;
    error = (error_foreground + error_background) * 100;
end

% Function to calculate scalar Gaussian
function prob_density = gaussian(x, u, sigma)
        prob_density = (1/sqrt(2*pi*sigma*sigma) * exp(-(x-u).
^2/(2*sigma.^2)));
end

% Function for z-test
function distance=z_test(var1, mu1, var2, mu2)
        diff = mu1-mu2;
        denom =  sqrt(var1+var2);
        distance = abs(diff/denom);
end
```