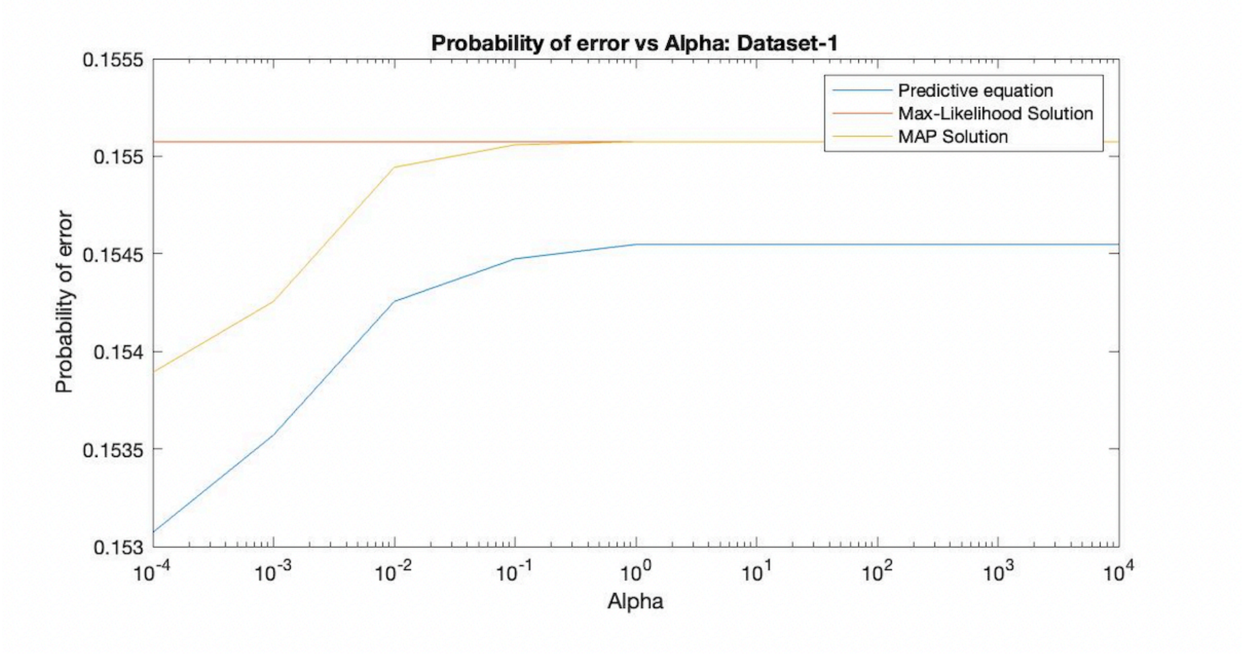
Srinidhi Bharadwaj Kalgundi Srinivas

A59010584

Assignment - 3

Attached below is the curve for probability of error as a function of alpha where alpha ranges from [1e-4, 1e4].



a) For Bayesian Parameter Estimation, mean of the posterior is governed by the covariance which in turn is governed by the alpha value. As the alpha value increases, mean of the posterior moves towards the mean of the MLE solution which is clearly observed from the graph below. This causes the probability of error to increase to a certain point and saturate after a certain value of alpha

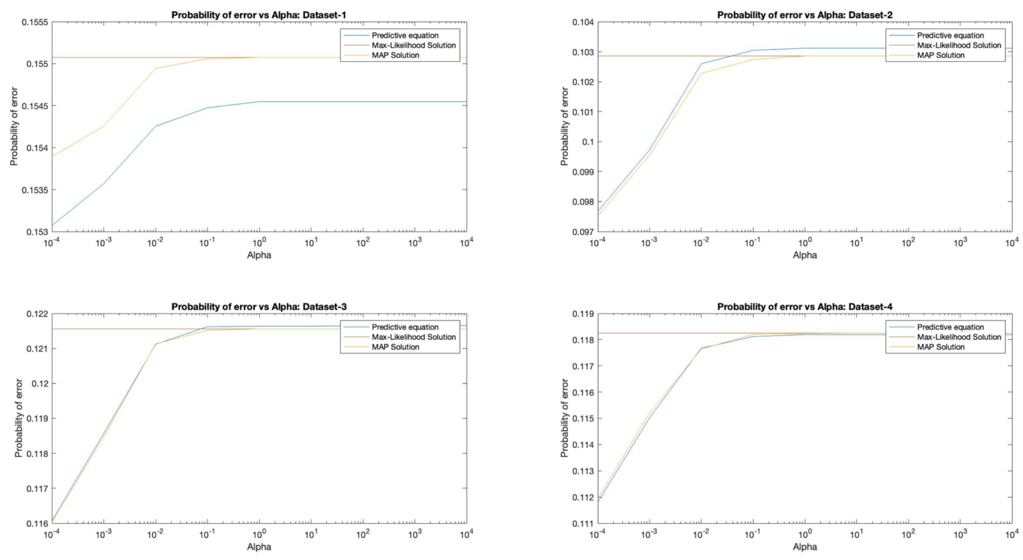
b) Graph for probability of error as a function of alpha is for ML estimates are as shown in Figure: 1. Graph is a straight line horizontail to the alpha axis as the ML estimates do not depend on the values of alpha and depends only on the mean and covariances of the dataset.

MAP estimate for the mean

c) Probability of errors as a function of alpha for the MAP estimates of the mean are shown in Figure: 1. Since MAP estimate shares the mean with Bayesian parameter estimation, probability of error is small when alpha is small and increase as alpha increases. Value of POE for MAP is slightly larger than that of the Bayesian parameter estimation as contribution of the prior covariance is smaller for MAP,

Quiz - d

Attached below is the curve for probability of error as a function of alpha for strategy 1 for D1, D2, D3 and D4 datasets

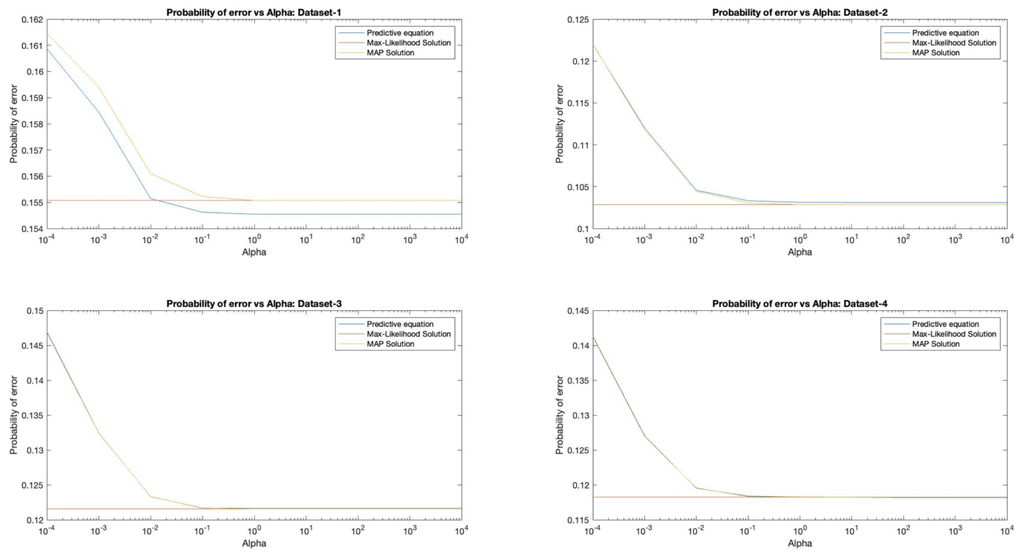


Change in behavior among different datasets:

It could be observed that in dataset one, Bayes predictive model is obviously performed better than ML and MAP. In dataset two, Bayes predictive model performed worse than ML and MAP. In datasets three and four, Bayes predictive model just performs slightly better than ML and MAP. This is because different datasets have different prior information for mean and the covariance. Size of datasets play an important role in determining the mean and covariances. As the size of the dataset increases, the mean and covariance estimation move towards the ML estimates.

e) Strategy-2

Attached below is the curve for probability of error as a function of alpha for strategy 2 for D1, D2, D3 and D4 datasets



Observation and explanation:

In both the strategies both the probabilities of error for MLE is fixed which is expected as the MLE estimates are only dependent on the data.

The probabilites of error in strategy 1 for Bayesian and MAP shows an increasing trend before saturating as the value of alpha increases. In Strategy 2, however, this trend is reversed and the probabilities of error reduces as alpha increases. However, after a certain value of alpha, the value is saturated and the saturation values for different datasets are the same in both the strategies. PoE is higher for smaller alpha in strategy 2 as shown in Figure 3. This trend is due to the fact that the prior information are different in both the strategies. When alpha is small, prior information has more weights. Strategy 2's prior information is worse as it provides the same mean value for both the classes whereas in strategy 1, different means are provided for foreground and background classes. This causes the Bayesian and MAP to be much worse off than MLE in strategy 2. As the alpha value increases, both Bayesian and MAP converge to MLE.

Type *Markdown* and LaTeX: α^2

Matlab Code:

```
clc;
clear;
close all;
% Load training samples and alpha

TrainingSet = load('hw3Data/TrainingSamplesDCT_subsets_8.mat')
;
dataset_bg_1 = TrainingSet.D1_BG;
dataset_fg_1 = TrainingSet.D1_FG;
dataset_bg_2 = TrainingSet.D2_BG;
dataset_fg_2 = TrainingSet.D2_FG;
dataset_bg_3 = TrainingSet.D3_BG;
dataset_fg_3 = TrainingSet.D3_FG;
dataset_bg_4 = TrainingSet.D4_BG;
dataset_fg_4 = TrainingSet.D4_FG;

dataset_bg_list = {dataset_bg_1, dataset_bg_2, dataset_bg_3, d
ataset_bg_4};
dataset_fg_list = {dataset_fg_1, dataset_fg_2, dataset_fg_3, d
ataset_fg_4};

% Load alphas
alpha = load('hw3Data/Alpha.mat');
alpha = alpha.alpha; % Assing the alpha structure to the varia
ble
num_alpha = size(alpha, 2);

% Load priors for 2 different strategies
strategy_1 = load("hw3Data/Prior_1.mat");
strategy_2 = load("hw3Data/Prior_2.mat");

strategy_list = [strategy_1, strategy_2];

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

```
cheetah_scale = im2double(cheetah);
cheetah_pad = zeros(size(cheetah_scale,1)+7,size(cheetah_scale
,2)+7);
cheetah_pad(3:257,3:272)=cheetah_scale;
cheetah = cheetah_pad;

[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

true_image = imread('../cheetah_mask.bmp');

for plan = 1:2 %Adding for scalability
    strategy = strategy_list(plan);
    figure;
    for data = 1:4 %Adding for scalability
        dataset_bg = dataset_bg_list(data);
        dataset_fg = dataset_fg_list(data);

        [bg_rows, bg_cols] = size(dataset_bg{1, 1});
        [fg_rows, fg_cols] = size(dataset_fg{1, 1});

        covar_bg = cov(dataset_bg{1, 1});
        covar_fg = cov(dataset_fg{1, 1});

        mean_bg = mean(dataset_bg{1, 1});
        mean_fg = mean(dataset_fg{1, 1});
        error = [];
        error_ML = [];
        error_MAP = [];
        subplot(2,2,data)
        set(gcf,'Position', get(0, 'ScreenSize'))

        for alpha_idx = 1:num_alpha
            prior_sigma_0 = zeros(64,64);
            for cov_idx=1:size(prior_sigma_0, 1)
                prior_sigma_0(cov_idx, cov_idx) = alpha(alpha_
idx) * strategy.W0(cov_idx);
```

```
            end

            prior_background = bg_rows/(bg_rows + fg_rows);
            prior_foreground = fg_rows/(fg_rows + bg_rows);

            % Bayesian BDR
            % Background parameter distribution
            inv_covar_bg = inv(covar_bg);
            covar_bg_inter = inv(inv(prior_sigma_0) + bg_rows*
inv(covar_bg));
            covar_bg = covar_bg_inter+covar_bg;
            inv_prior = inv(prior_sigma_0);

            mu_background_bayes = covar_bg_inter * (inv_prior*
transpose(strategy.mu0_BG) + bg_rows*inv_covar_bg * transpose(
mean_bg));
            wi_background = inv(covar_bg);
            determinant_background = det(covar_bg);

            % Foreground parameter distribution
            inv_covar_fg = inv(covar_fg);
            covar_fg_inter = (inv(prior_sigma_0) + fg_rows*inv
(covar_fg));
            covar_fg_inter = inv(covar_fg_inter);
            covar_fg = covar_fg_inter+covar_fg;
            inv_prior = inv(prior_sigma_0);

            mu_foreground_bayes = covar_fg_inter * (inv_prior*
transpose(strategy.mu0_FG) + fg_rows*inv_covar_fg * transpose(
mean_fg));
            wi_foreground = inv(covar_fg);
            determinant_foreground = det(covar_fg);

            result_64 = zeros(unpadded_rows, unpadded_cols);

            % 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_foregroun
d_bayes) * wi_foreground * (x-mu_foreground_bayes);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_foreground));
                    p_y_x_cheetah = exp_coeff * exp(exp_func)
* prior_foreground;

                    % Grass
                    exp_func = -0.5 * transpose(x-mu_backgroun
d_bayes) * wi_background * (x-mu_background_bayes);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_background));
                    p_y_x_grass = exp_coeff * exp(exp_func) *
prior_background;

                    if(p_y_x_cheetah > p_y_x_grass)
                        result_64(i, j) = 1;
                    end
                end
            end
%             figure;
%             imagesc(result_64);
%             title('Prediction with 64 dimensions')
%             colormap(gray(255));
            error_tmp = calc_error(true_image, result_64, prio
r_foreground, prior_background);
            error = [error error_tmp];


            % Maximum Likelihood - BDR
            covar_bg = cov(dataset_bg{1, 1});
            covar_fg = cov(dataset_fg{1, 1});

            mean_bg = mean(dataset_bg{1, 1});
            mean_fg = mean(dataset_fg{1, 1});

            result_64 = zeros(unpadded_rows, unpadded_cols);
            mu_foreground = transpose(mean_fg);
            mu_background = transpose(mean_bg);
            determinant_foreground = det(covar_fg);
            determinant_background = det(covar_bg);
            wi_foreground = inv(covar_fg);
            wi_background = inv(covar_bg);

            % 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_foregroun
d) * wi_foreground * (x-mu_foreground);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_foreground));
                    p_y_x_cheetah = exp_coeff * exp(exp_func)
* prior_foreground;

                    % Grass
                    exp_func = -0.5 * transpose(x-mu_backgroun
d) * wi_background * (x-mu_background);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_background));
                    p_y_x_grass = exp_coeff * exp(exp_func) *
prior_background;

                    if(p_y_x_cheetah > p_y_x_grass)
                        result_64(i, j) = 1;
                    end
                end
            end
            error_tmp = calc_error(true_image, result_64, prio
r_foreground, prior_background);
            error_ML = [error_ML error_tmp];

            % MAP - BDR
            result_64 = zeros(unpadded_rows, unpadded_cols);
            mu_background = mu_background_bayes;
            mu_foreground = mu_foreground_bayes;

            % 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_foregroun
d) * wi_foreground * (x-mu_foreground);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_foreground));
                    p_y_x_cheetah = exp_coeff * exp(exp_func)
* prior_foreground;

                    % Grass
                    exp_func = -0.5 * transpose(x-mu_backgroun
d) * wi_background * (x-mu_background);
                    exp_coeff = 1/(sqrt(((2*pi)^64) * determin
ant_background));
                    p_y_x_grass = exp_coeff * exp(exp_func) *
prior_background;

                    if(p_y_x_cheetah > p_y_x_grass)
                        result_64(i, j) = 1;
                    end
                end
            end
            error_tmp = calc_error(true_image, result_64, prio
r_foreground, prior_background);
            error_MAP = [error_MAP error_tmp];
        end

        plot(alpha,error, alpha, error_ML, alpha, error_MAP);
        legend('Predictive equation','Max-Likelihood Solution'
,'MAP Solution');
        set(gca, 'XScale', 'log');
        title(['Probability of error vs Alpha: Dataset-', num2
str(data)]);
        xlabel('Alpha');
        ylabel('Probability of error');


    end


end


%------------------------- Helper functions ------------------
------------%

% 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);
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
```

In [ ]: