ECE 271A - Statistical Learning I - Assignment 3

Saqib Azim (A59010162)
UC San Diego
sazim@ucsd.edu

Nov 22, 2021

1 Segmentation of Cheetah Image into cheetah (foreground) and ground (background)

Figure 1: Original Cheetah Image

Figure 2: GT Segmentation Mask

1.1 Data-Subset Information

Number of Samples in Dataset D1 (FG, BG) - 75, 300 $P_Y(FG) = 0.2$, $P_Y(BG) = 0.8$
Number of Samples in Dataset D2 (FG, BG) - 125, 500 $P_Y(FG) = 0.2$, $P_Y(BG) = 0.8$
Number of Samples in Dataset D3 (FG, BG) - 175, 700 $P_Y(FG) = 0.2$, $P_Y(BG) = 0.8$
Number of Samples in Dataset D4 (FG, BG) - 225, 900 $P_Y(FG) = 0.2$, $P_Y(BG) = 0.8$

1.2 Probability of Error Formula

$$P(E) = E_Y(P_{X|Y}(g(x) \neq i)|i) \tag{1}$$

$$P(E) = P_{X|Y}(g(x) \neq \text{FG}|\text{FG})P_Y(\text{FG}) + P_{X|Y}(g(x) \neq \text{BG}|\text{BG})P_Y(\text{BG}) \tag{2}$$

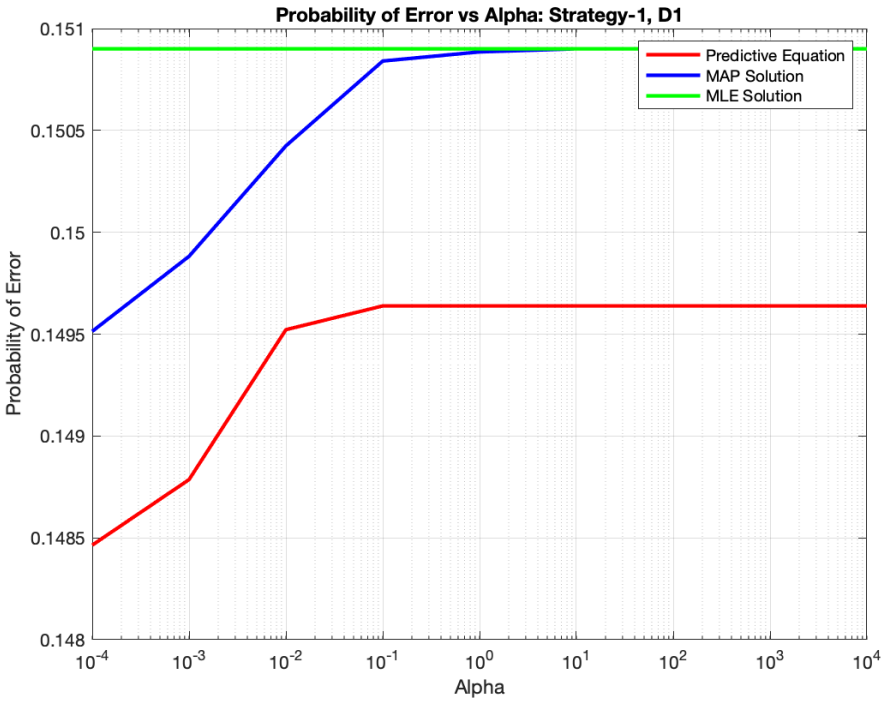

Figure 3: Probability of Error wrt Alpha - Strategy-1, D1

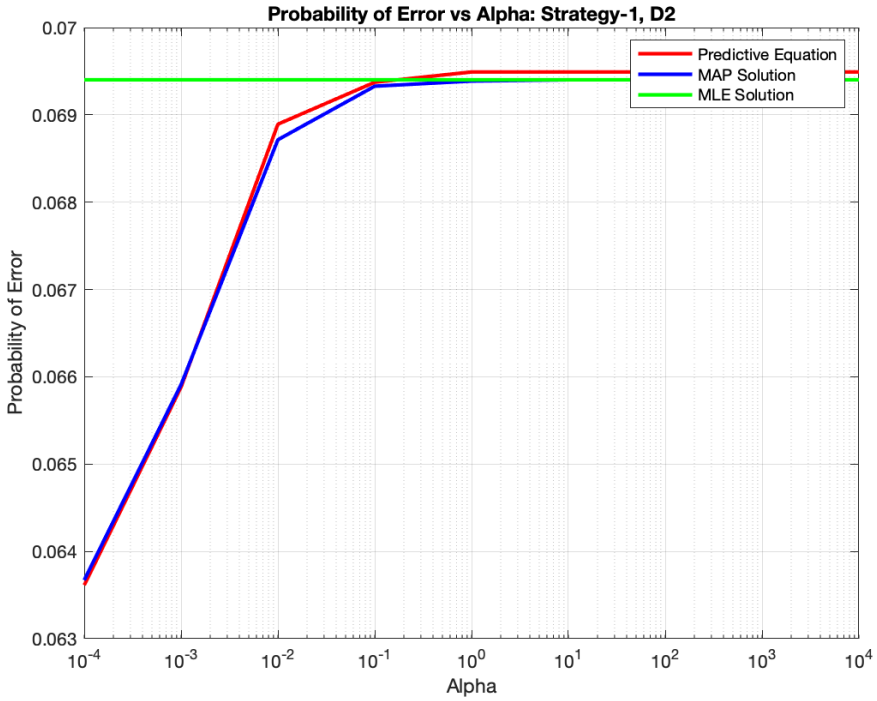

Figure 4: Probability of Error wrt Alpha - Strategy-1, D2

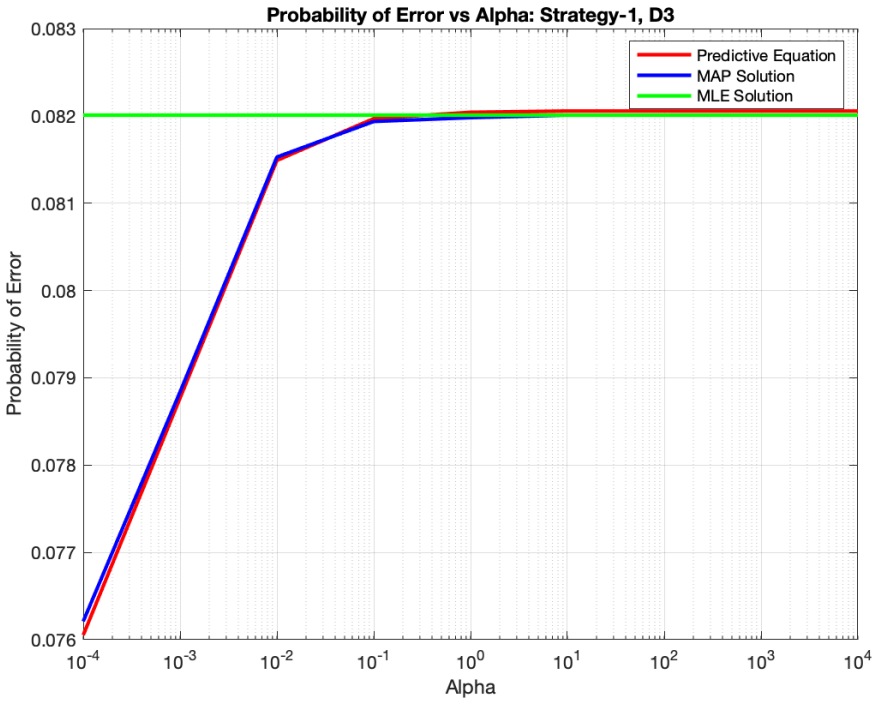

Figure 5: Probability of Error wrt Alpha - Strategy-1, D3

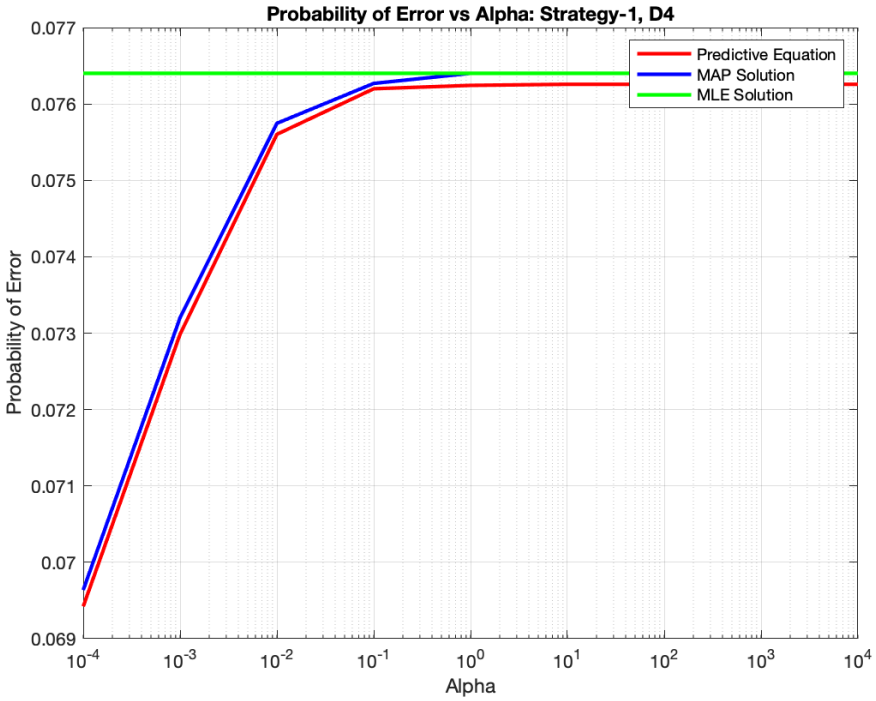

Figure 6: Probability of Error wrt Alpha - Strategy-1, D4

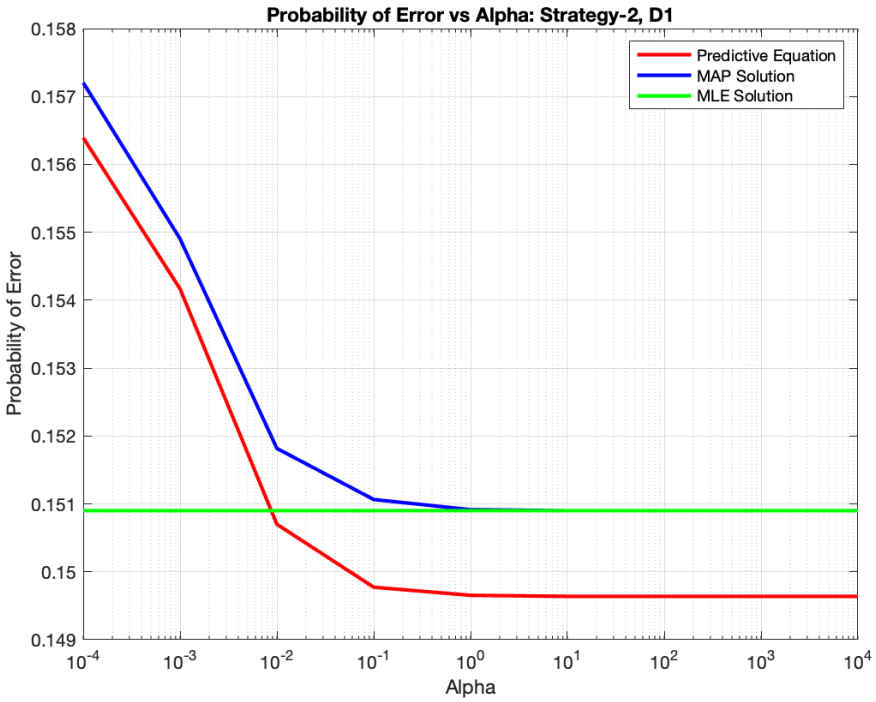

Figure 7: Probability of Error wrt Alpha - Strategy-2, D1

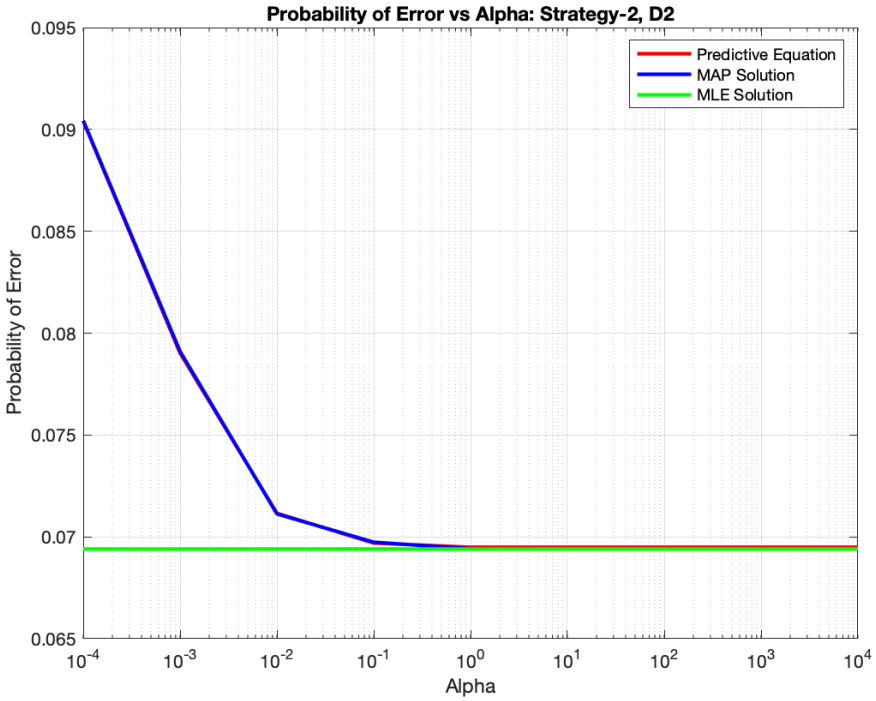

Figure 8: Probability of Error wrt Alpha - Strategy-2, D2

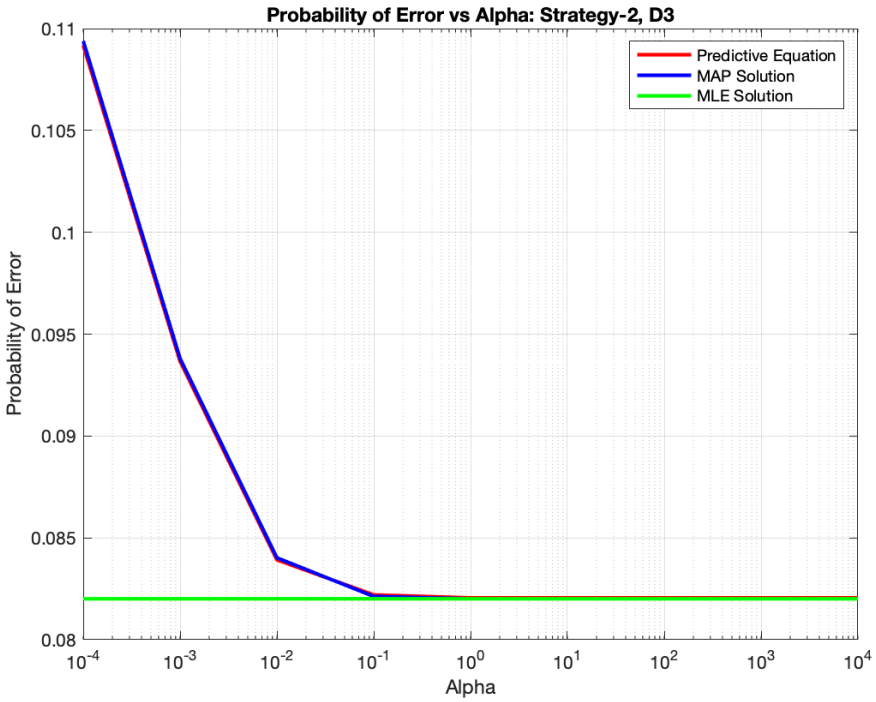

Figure 9: Probability of Error wrt Alpha - Strategy-2, D3

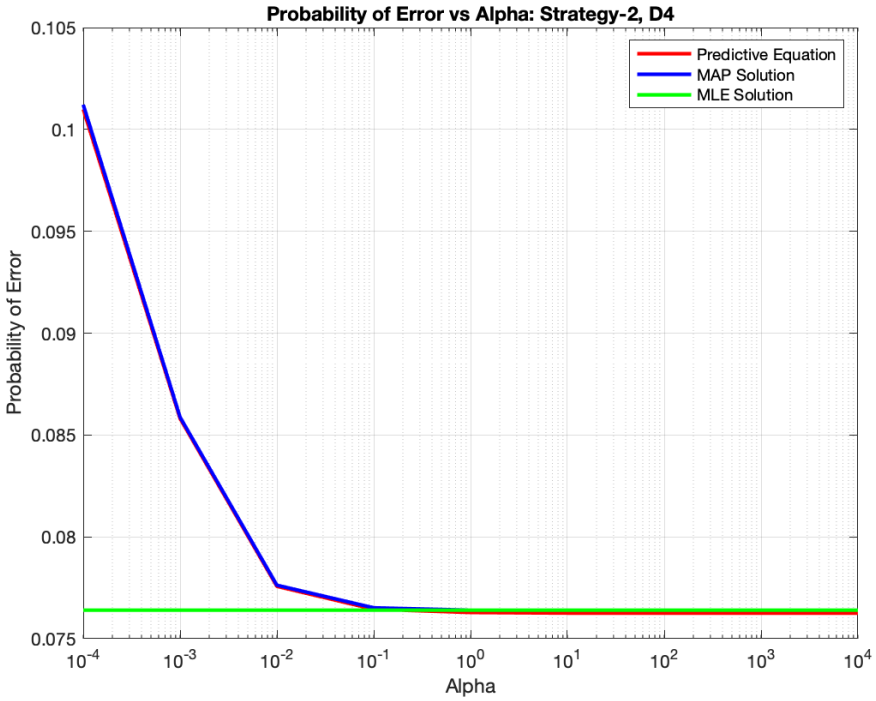

Figure 10: Probability of Error wrt Alpha - Strategy-2, D4

Bayesian Model for class-conditional

$$p_{x|\mu,\Sigma} = G(x, \mu, \Sigma_i)$$

Σ_i = sample covariance matrix $= \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)^T$

Assumption: Mean μ is unknown. $\mu \in R^{64}$

Mean Prior: $p_\mu(\mu) = G(\mu, \mu_0, \Sigma_0)$

where. $(\Sigma_0)_{ii} = \alpha w_i$ $\quad \Sigma_0 = \text{diag}(\alpha w_i)$

Strategy 1: $\mu_0^{FG}[0] = 1$ $\quad \mu_0^{BG}[0] = 3$

$\mu_0^{FG}[1:\text{end}] = 0$ $\quad \mu_0^{BG}[1:\text{end}] = 0$

Strategy 2: $\mu_0^{FG}[0] = 2$ $\quad \mu_0^{BG}[0] = 2$

$\mu_0^{FG}[1:\text{end}] = 0$ $\quad \mu_0^{BG}[1:\text{end}] = 0$

Bayesian-BDR

~~Risk~~ $i^*(x) = \underset{i}{\text{argmax}}\ p_{x|y,T}(x|i, D_i)\ P_y(i)$

$$p_{x|y,T}(x|i, D_i) = \int p_{x|\theta,y,T}(x|\theta, i, D_i)\ p_{\theta|y,T}(\theta|i, D_i)\, d\theta$$

drop class y ↓

$$= \int p_{x|\theta,y}(x|\theta, i)\ p_{\theta|y,T}(\theta|i, D_i)\, d\theta$$

$$p_{x|T}(x|D) = \int p_{x|\theta}(x|\theta)\ p_{\theta|T}(\theta|D)\, d\theta \leftarrow \text{Predictive Distribution}$$

$p_{x|\theta}(x|\theta)$ = probability distribution for observations given parameter θ.

$$P_{\mu|T}(\mu|D) = \frac{P_{T|\mu}(D|\mu)\, P_{\mu}(\mu)}{P_T(D)} \propto P_{T|\mu}(D|\mu)\, P_{\mu}(\mu) \quad (2)$$

$$P_{\mu|T}(\mu|D) \propto P_{x_1, x_2 \ldots x_m|\mu}(x_1, x_2 \ldots x_m|\mu)\, P_{\mu}(\mu)$$

$x_1, x_2 \ldots x_m$ are IID samples.

$$P_{\mu|T}(\mu|D) \propto \prod_{j=1}^{m} P_{x_j|\mu}(x_j|\mu)\, P_{\mu}(\mu)$$

$$P_{\mu|T}(\mu|D) \propto \prod_{j=1}^{m} \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left[-\frac{1}{2}(x_j-\mu)^T \Sigma^{-1}(x_j-\mu)\right].$$

$$\frac{1}{\sqrt{(2\pi)^d |\Sigma_0|}} \exp\left[-\frac{1}{2}(\mu-\mu_0)^T \Sigma_0^{-1}(\mu-\mu_0)\right]$$

Let $k = \dfrac{1}{(2\pi)^{\frac{d}{2}(m+1)} |\Sigma|^{m/2} |\Sigma_0|^{\frac{1}{2}}}$

$$P_{\mu|T}(\mu|D) \propto k \exp\left[\sum_{j=1}^{m} -\frac{1}{2}(x_j-\mu)^T \Sigma^{-1}(x_j-\mu) - \frac{1}{2}(\mu-\mu_0)^T \Sigma_0^{-1}(\mu-\mu_0)\right]$$

$$\propto k \exp\left[-\frac{1}{2}\left\{\sum_{j=1}^{m} x_j^T \Sigma^{-1} x_j - 2\sum_{j=1}^{m} x_j^T \Sigma^{-1}\mu + m\mu^T\Sigma^{-1}\mu + \mu^T\Sigma_0^{-1}\mu - 2\mu_0^T\Sigma_0^{-1}\mu + \mu_0^T\Sigma_0^{-1}\mu_0\right]\right.$$

$$\propto k \exp\left[\frac{1}{2}\left\{\mu^T(m\Sigma^{-1}+\Sigma_0^{-1})\mu - 2\left(\sum_{j=1}^{m} x_j^T\Sigma^{-1} + \mu_0^T\Sigma_0^{-1}\right)\mu + \left(\sum_{j=1}^{m} x_j^T\Sigma^{-1}x_j + \mu_0^T\Sigma_0^{-1}\mu_0\right)\right]\right.$$

Since $P_{\mu|T}(\mu|D) \sim G(\mu, \mu_1, \Sigma_1)$

$$\propto \exp\left[-\frac{1}{2}\left(\mu^T\Sigma_1^{-1}\mu - 2\mu_1^T\Sigma_1^{-1}\mu + \mu_1^T\Sigma_1^{-1}\mu_1\right)\right]$$

Thus, $\mu^T \Sigma_1^{-1} \mu = \mu^T \left(m\Sigma^{-1} + \Sigma_0^{-1} \right) \mu$

$$\Sigma_1^{-1} = \left(m\Sigma^{-1} + \Sigma_0^{-1} \right) \Rightarrow \quad \Sigma_1 = \left(m\Sigma^{-1} + \Sigma_0^{-1} \right)^{-1}$$

$$\mu_1^T \Sigma_1^{-1} \mu = \left(\sum_{j=1}^{m} x_j^T \Sigma^{-1} + \mu_0^T \Sigma_0^{-1} \right) \mu$$

$$\mu_1^T = \left(\sum_{j=1}^{m} x_j^T \Sigma^{-1} + \mu_0^T \Sigma_0^{-1} \right) \Sigma_1$$

$$\mu_1 = \Sigma_1 \left[\Sigma^{-1} (m\mu_{ML}) + \Sigma_0^{-1} \mu_0 \right]$$

$$\boxed{\begin{aligned} \mu_1 &= \Sigma_1 \left[m\Sigma^{-1} \mu_{ML} + \Sigma_0^{-1} \mu_0 \right] \\ \Sigma_1 &= \left(m\Sigma^{-1} + \Sigma_0^{-1} \right)^{-1} \end{aligned}}$$

Posterior Mean and Covariance for the unknown mean μ.

m → number of samples.

Σ → Model sample covariance matrix

Posterior Probability distribution for unknown parameter μ.

$$p_{\mu|T}(\mu|D) = G(\mu, \mu_1, \Sigma_1)$$

$$= \frac{1}{\sqrt{(2\pi)^d |\Sigma_1|}} \exp\left[\frac{-1}{2} (x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1) \right].$$

Now, predictive distribution $p_{x|T}(x|D) = \int p_{x|\mu}(x|\mu)\, p_{\mu|T}(\mu|D)\, d\mu.$

$$p_{x|\mu}(x|\mu) = G(x; \mu, \Sigma)$$

$$p_{x|T}(x|D) = \int G(x, \mu, \Sigma) \cdot G(\mu, \mu_1, \Sigma_1)\, d\mu.$$

where Σ, μ_1, Σ_1 are known quantities.

$$p_{X|T}(x|D) = \int \underbrace{G(x-\mu, 0, \Sigma)}_{f(x-\mu)} \cdot \underbrace{G(\mu, \mu_1, \Sigma_1)}_{h(\mu)} d\mu$$

$$= \int f(x-\mu)\, h(\mu)\, d\mu \qquad f(x) = G(x, 0, \Sigma)$$
$$h(x) = G(x, \mu_1, \Sigma_1)$$

$$= G(x, 0, \Sigma) * G(x, \mu_1, \Sigma_1)$$

$*$ ← convolution operator.

$$\boxed{p_{X|T}(x|D) = G(x, \mu_1, \Sigma + \Sigma_1)}.$$

For MAP estimate of μ.

$$\mu_{MAP} = \underset{\mu}{\text{argmax}}\; p_{\mu|T}(\mu|D_1) = \underset{\mu}{\text{argmax}}\; G(\mu, \mu_1, \Sigma_1) = \mu_1$$

$$p_{X|T}(x|D_1) = p_{X|\mu}(x|\mu_{MAP}) = p_{X|\mu}(x|\mu_1)$$

$$= G(x, \mu_1, \Sigma)$$

μ_1 ← mean of posterior distribution of μ.

Σ ← sample covariance matrix

1.3 Explanation Q.a: Bayesian Solution vs Alpha Plot

As the value of α increases, the probability of error $P(E)$ for Bayesian solution also increases and becomes constant for higher values of α.

$$\Sigma_0 = \text{ diag } (\alpha W_i) = \alpha \text{ diag } (W_i) \tag{3}$$

α is increased from $1e^{-4}$ to $1e^4$ in powers of 10.

$$\Sigma_0 = \alpha \text{ diag } (W_i) = \alpha K \quad \text{where} \quad K = \text{ diag } (W_i) = \text{constant} \tag{4}$$

Covariance Matrix of the posterior dist. of mean parameter

$$\Sigma_1 = (m\Sigma^{-1} + \Sigma_0^{-1})^{-1} = (m\Sigma^{-1} + \frac{1}{\alpha}K^{-1})^{-1} \tag{5}$$

$\frac{1}{\alpha}$ acts as a relative weight between the sample covariance inverse (Σ^{-1}) and the prior covariance inverse (K^{-1}). As α increases and tends towards ∞, $\frac{1}{\alpha}$ goes to 0, and

$$\Sigma_1 = \frac{1}{m}\Sigma \quad \text{as} \quad \alpha \to \infty \quad \text{and} \quad \Sigma_1 = \alpha K = \Sigma_0 \quad \text{as} \quad \alpha \to 0^+ \tag{6}$$

$$\mu_1 = \Sigma_1(m\Sigma^{-1}\mu_{ML} + \Sigma_0^{-1}\mu_0) = \Sigma_1(m\Sigma^{-1}\mu_{ML} + \frac{1}{\alpha}K^{-1}\mu_0) \tag{7}$$

$$\mu_1 = \mu_{ML} \quad \text{as} \quad \alpha \to \infty \quad \text{and} \quad \mu_1 = \mu_0 \quad \text{as} \quad \alpha \to 0^+ \tag{8}$$

For Bayesian solution, the predictive distribution is

$$P_{X|T}(x|D) = G(x, \mu_1, \Sigma + \Sigma_1) \tag{9}$$

whereas for the MAP solution, the predictive distribution is

$$P_{X|T}(x|D) = G(x, \mu_1, \Sigma) \tag{10}$$

As α increases from 0 to ∞, the mean and covariance of the posterior distribution of the unknown parameter μ goes from the mean and covariance of the prior distribution towards the mean and covariance of the MLE solution. For all the datasets under strategy-1, the prob. of error for the Bayesian and MAP solution increases as α is increased from 0.0001 to 0.1 or 1. This means that when the posterior of the mean shifts away from the mean prior distribution and moves towards the data-based MLE solution, the prob. of error increases. Therefore, for strategy-1, lower values of α provides better results (in terms of prob. of error) and signifies that the prior assumption regarding the unknown parameter μ has better classification accuracy as compared to the data based MLE solution. That is for strategy-1, the prior is better than the training data for both the Bayesian and MAP solution and for all the datasets.

As α increases from 1 to 10000, the prob. of error becomes more or less constant for both the Bayesian and MAP solution. For dataset D2, D3, and D4, the Bayesian and MAP solution approximately converge to the MLE solution with higher values of alpha whereas for dataset

D1, only the MAP solution converges to the MLE solution while the Bayesian solution has still lower prob. of error compared to MLE solution. This is because in the low-data regime, the Bayesian is always better than the MLE solution. As the data increases, the MLE solution becomes more prominent and relevant, and the Bayesian and MAP solution converges to the MLE solution.

As alpha increases, the diagonal elements of the prior covariance matrix Σ_0 increases. The diagonal elements corresponds to the variance of the individual elements of prior of the mean distribution. For gaussian mean prior and gaussian model distribution, the precision of the Bayesian solution is related to the precision of MLE solution and prior belief as follows:

$$P_{\text{Bayes}} = P_{\text{MLE}} + P_{\text{Prior}} \tag{11}$$

As the alpha increases, the variances of individual components of prior belief (diagonal elements) increases, and thus the overall precision of the prior belief decreases. This results in decrease of precision of Bayesian solution which means the accuracy of Bayesian solution decreases. Thus, the prob. of error increases as the value of alpha increases.

1.4 Explanation Q.b: ML vs Bayesian Solution Prob. of Error

In Figure 3, 4, 5, 6, since the dataset D1 has relatively small number of training samples compared to D2, D3, and D4, the Bayesian solution has lower prob. of error compared to ML solution for all values of alpha. In addition, the Bayesian solution converges towards the ML solution as α increases (explained above in subsection 1.3).

1.5 Explanation Q.c: MAP vs Bayesian vs MLE

In all the plots, the MAP solution follows similar trajectory as the Bayesian solution for dataset D1 (with an offset) and almost the same trajectory as the Bayesian solution for dataset D2, D3, and D4. The predictive distribution of both the Bayesian and MAP solution are as follows:

$$\text{MAP solution:} \quad P_{X|T}(x|D) = G(x, \mu_1, \Sigma) \tag{12}$$

$$\text{Bayesian Solution:} \quad P_{X|T}(x|D) = G(x, \mu_1, \Sigma + \Sigma_1) \tag{13}$$

The Bayesian-BDR and MAP-BDR are only different because of the difference in the covariance matrix of the predictive distribution of the two solutions. The mean of the predictive gaussian distribution is same for both the MAP and Bayesian solution. The covariance of the predictive distribution of the Bayesian solution is $\Sigma + \Sigma_1$ whereas for MAP solution is Σ. As number of training sample increases in D2, D3, and D4, Σ_1 moves closer to the sample covariance matrix Σ. Therefore, both the solutions produce approx. the same prob. of error.

1.6 Explanation Q.d: Effect of increased training sample (D1-D4) on the solutions

Datasets D1, D2, D3, D4 have progressively larger number of training samples but the prior probability of FG and BG class remains the same for all the datasets. As the number

of training samples increases, the Bayesian solution converges to the MLE solution since the posterior distribution becomes narrower. Therefore, the Bayesian estimate has more confidence on what the parameter value is and less uncertainty on the model parameters. In addition, with large number of training samples, the MLE solution becomes more relevant. For both strategies, as the dataset size increases, the converges towards the MLE solution. From the above precision equation, the precision of Bayesian solution is generally higher than the precision of MLE solution. For relatively small amount of data (such as D1), the Bayesian solution is more relevant and performs much better than MLE solution. MLE solution is good for relatively large datasets (such as D4). In case of small data, parameter posterior dist. has lot of variation. Using ML solution results in discarding a lot of information whereas Bayesian solution provides complete characterization of the posterior parameters.

1.7 Explanation Q.e: Effect of changing from strategy-1 to strategy-2

In strategy-1, the prior mean of the unknown parameter μ is different for both the classes. The prior mean is also the mean of the predictive gaussian distribution for the Bayesian and MAP solution. Thus, it results in two distinct Gaussians with separate means for the FG and BG class. In strategy-2, the mean of the prior distribution of the parameter μ is same for both the FG and BG class. As shown earlier, when α increases from 0 to ∞, the posterior mean tends to move from the prior mean to the MLE-based mean. When α is close to 0, the posterior mean tends towards μ_0 (same for both classes). Thus, the mean vector of the predictive distributions for both the Bayesian and MAP solutions have the same mean for FG and BG class. That is, the two gaussians overlap each other and therefore, it becomes difficult to classify the two classes using Bayesian and MAP solutions. Hence the prob. of error is higher for Bayesian and MAP solution for low alpha values. As alpha increases, the posterior mean tends to shift away from the prior and move towards the MLE solution. Therefore, the gaussian distributions for the two classes separate and becomes easier to classify FG and BG class leading to decrease in the prob. of error.

2 Appendix

2.1 Matlab Code - Main Code: 'main.m'

```
clc; clear;
close all;

% load the original image and ground-truth segmentation mask
img = im2double(imread("../data/cheetah.bmp"));
img = img(:, 1:end-2);
seg_mask_gt = im2double(imread('../data/cheetah_mask.bmp'));
seg_mask_gt = seg_mask_gt(:, 1:end-2);

% load the zigzag pattern file
```

```
zigzag_pat = importdata("../data/zigzag_pattern.txt");
zigzag_pat_lin = zigzag_pat(:) + 1;   % adding 1 for converting to matlab indexes

% load the training sample DCT matrix
TS_DCT = load("../data/TrainingSamplesDCT_8_new.mat");
TS_DCT_FG = TS_DCT.TrainsampleDCT_FG;
TS_DCT_BG = TS_DCT.TrainsampleDCT_BG;

% load the subset training samples DCT matrix
TS_DCT_subsets = load("../data/TrainingSamplesDCT_subsets_8.mat");

% get the D1 data
D1_FG = TS_DCT_subsets.D1_FG;
D1_BG = TS_DCT_subsets.D1_BG;

% get the D2 data
D2_FG = TS_DCT_subsets.D2_FG;
D2_BG = TS_DCT_subsets.D2_BG;

% get the D3 data
D3_FG = TS_DCT_subsets.D3_FG;
D3_BG = TS_DCT_subsets.D3_BG;

% get the D4 data
D4_FG = TS_DCT_subsets.D4_FG;
D4_BG = TS_DCT_subsets.D4_BG;

for strategy = 1:2
   for dataset = 1:4
% strategy = 1;
% dataset = 1;

if (dataset == 1)
    D_FG = D1_FG;
    D_BG = D1_BG;
elseif (dataset == 2)
    D_FG = D2_FG;
    D_BG = D2_BG;
elseif (dataset == 3)
    D_FG = D3_FG;
    D_BG = D3_BG;
elseif (dataset == 4)
    D_FG = D4_FG;
    D_BG = D4_BG;
end
```

```
% load alpha.mat and compute parameters for mean prior dist.
mu_prior_sigma_alpha = load("../data/Alpha.mat").alpha;
num_alpha_val = size(mu_prior_sigma_alpha, 2);

prob_error_1 = zeros(1, num_alpha_val);
prob_error_2 = zeros(1, num_alpha_val);

prob_error_1_MAP = zeros(1, num_alpha_val);
prob_error_2_MAP = zeros(1, num_alpha_val);

for k = 1:num_alpha_val
    alpha = mu_prior_sigma_alpha(1, k);

    fprintf("\n");
    disp("Strategy - "+strategy+"  ,  Dataset - D"+dataset+"  ,  Alpha: "+alpha);

    [prob_error_1(1,k), prob_error_2(1,k)] = classify_FG_BG_Bayesian(img, seg_mask_gt, z

    [prob_error_1_MAP(1,k), prob_error_2_MAP(1,k)] = classify_FG_BG_MAP(img, seg_mask_gt
end

[prob_error_1_MLE, prob_error_2_MLE] = classify_FG_BG_MLE(img, seg_mask_gt, zigzag_pat_l
% disp("Prob. of Error (MLE, Method-1): "+ prob_error_1_MLE);
% disp("Prob. of Error (MLE, Method-2): "+ prob_error_2_MLE);
prob_error_1_MLE_vec = ones(1, num_alpha_val)*prob_error_1_MLE;

lw = 2;
figure;
plot(mu_prior_sigma_alpha, prob_error_1, 'color', 'r', 'LineWidth', lw);
hold on;
plot(mu_prior_sigma_alpha, prob_error_1_MAP, 'color', 'b', 'LineWidth', lw);
hold on;
plot(mu_prior_sigma_alpha, prob_error_1_MLE_vec, 'color', 'g', 'LineWidth', lw);

set(gca, 'XScale', 'log');
ax.FontSize = 25;
xlabel("Alpha");
ylabel("Probability of Error");
title("Probability of Error vs Alpha: Strategy-"+strategy+", D"+dataset);
legend('Predictive Equation', 'MAP Solution', 'MLE Solution');
grid on;
% close all;
saveas(gcf, "../plots/S"+strategy+"_D"+dataset+"_prob_err_plot.png");
```

```
close;
   end
end
```

2.2 Matlab Code - Function 'classify_FG_BG_Bayesian.m'

```
function [prob_error_1, prob_error_2] = classify_FG_BG_Bayesian(img, seg_mask_gt, zigzag
    [img_height, img_width] = size(img);

    % number of training samples in Di for FG and BG
    num_TS_FG = size(TS_FG, 1);
    num_TS_BG = size(TS_BG, 1);

    if strategy == 1
        % load prior1.mat
        prior1 = load("../data/Prior_1.mat");
        mu_prior_mu_FG = prior1.mu0_FG;
        mu_prior_mu_BG = prior1.mu0_BG;
        mu_prior_sigma_weights = prior1.W0;

    elseif strategy == 2
        % load prior2.mat
        prior2 = load("../data/Prior_2.mat");
        mu_prior_mu_FG = prior2.mu0_FG;
        mu_prior_mu_BG = prior2.mu0_BG;
        mu_prior_sigma_weights = prior2.W0;
    end

    mu_prior_sigma = alpha * diag(mu_prior_sigma_weights);   % same for FG and BG classe
    mu_prior_sigma_inv = inv(mu_prior_sigma);

    % compute the covariance matrix of X|Y,T equal to the sample covariance
    TS_mu_FG = mean(TS_FG);  % 1 x 64
    TS_FG_bar = TS_FG - TS_mu_FG;   % num_sample x 64
    TS_sigma_FG = (transpose(TS_FG_bar) * TS_FG_bar) / num_TS_FG;
%     TS_sigma_FG = compute_cov_matrix(TS_FG, TS_mu_FG);
    TS_sigma_FG_inv = inv(TS_sigma_FG);

    TS_mu_BG = mean(TS_BG);
    TS_BG_bar = TS_BG - TS_mu_BG;
    TS_sigma_BG = (transpose(TS_BG_bar) * TS_BG_bar) / num_TS_BG;
%     TS_sigma_BG = compute_cov_matrix(TS_BG, TS_mu_BG);
    TS_sigma_BG_inv = inv(TS_sigma_BG);

    % compute posterior mean and covariance for FG class
```

```
mu_post_sigma_FG = inv(num_TS_FG * TS_sigma_FG_inv + mu_prior_sigma_inv);
mu_post_mu_FG = transpose((num_TS_FG * TS_mu_FG * TS_sigma_FG_inv + mu_prior_mu_FG *

% compute posterior mean and covariance for BG class
mu_post_sigma_BG = inv(num_TS_BG * TS_sigma_BG_inv + mu_prior_sigma_inv);
mu_post_mu_BG = transpose((num_TS_BG * TS_mu_BG * TS_sigma_BG_inv + mu_prior_mu_BG *

% compute parameters of predictive distribution for FG and BG class
mu_pred_FG = mu_post_mu_FG;
mu_pred_BG = mu_post_mu_BG;

sigma_pred_FG = mu_post_sigma_FG + TS_sigma_FG;
sigma_pred_FG_inv = inv(sigma_pred_FG);

sigma_pred_BG = mu_post_sigma_BG + TS_sigma_BG;
sigma_pred_BG_inv = inv(sigma_pred_BG);

% ML-estimate for class prior PY_FG and PY_BG
num_TS = num_TS_FG + num_TS_BG;
PY_FG = num_TS_FG / num_TS;   % P(Y = FG)
PY_BG = num_TS_BG / num_TS;   % P(Y = BG)

% classification of each pixel into FG and BG
d = 64;                           % feature dimension
block_dct_vec = zeros(d, 1);
seg_mask_res = zeros(size(img));

PYGX_FG_x = zeros(size(img));
PYGX_BG_x = zeros(size(img));

% pad test image with 7 layers to the right and bottom
img_pad = img_padding(img);

% predict cheetah image using BDR
norm_const_FG = 1 / sqrt(power(2 * pi, d) * det(sigma_pred_FG));
norm_const_BG = 1 / sqrt(power(2 * pi, d) * det(sigma_pred_BG));

for i = 1:img_height
    for j = 1:img_width

        img_block_dct = dct2(img_pad(i:i+7,j:j+7));
        block_dct_vec(zigzag_pat_lin, 1) = img_block_dct(:);

        x_minus_mu_FG = block_dct_vec - mu_pred_FG;
        x_minus_mu_BG = block_dct_vec - mu_pred_BG;
```

```
            PXGYT_x_FG = norm_const_FG * exp(-0.5 * transpose(x_minus_mu_FG) * sigma_pre
            PXGYT_x_BG = norm_const_BG * exp(-0.5 * transpose(x_minus_mu_BG) * sigma_pre
            PX_x = PXGYT_x_FG * PY_FG + PXGYT_x_BG * PY_BG;

            PYGX_FG_x(i,j) = (PXGYT_x_FG * PY_FG) / PX_x;
            PYGX_BG_x(i,j) = (PXGYT_x_BG * PY_BG) / PX_x;

            if (PYGX_FG_x(i,j) > PYGX_BG_x(i,j))
                seg_mask_res(i,j) = 1;
            else
                seg_mask_res(i,j) = 0;
            end
        end
    end

    % compute prob of error for the Bayesian solution
    prob_error_1 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 1);
    prob_error_2 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 2);
    disp("Probability of Error (Predictive): "+prob_error_1);

%   whos;
%     f = figure();
%     ax = gca;
%     ax.FontSize = 16;
%     f.WindowState = 'maximized';
%     subplot(2,2,1);
%     imshow(seg_mask_gt);
%     title('Ground-Truth Segmentation Mask')

%     subplot(2,2,2);
%     imshow(seg_mask_res);
%     title('Segmentation Result')
    % imwrite(seg_mask_res_64, "../plots/seg_mask_res_64.png");

%     subplot(2,2,3);
%     imshow(img);
%     title('Original Image')
    % imwrite(seg_mask_res_8, "../plots/seg_mask_res_best_8.png");

%     subplot(2,2,4);
%     imshow(img_pad);
%     title('Padded Image')
    % imwrite(seg_mask_res_8, "../plots/seg_mask_res_best_8.png");
```

```
    imwrite(seg_mask_res, "../plots/S"+strategy+"_D"+dataset+"/seg_mask_res_alpha_"+alph
    close all;
end
```

2.3 Matlab Code - Function 'classify_FG_BG_MLE.m'

```
function [prob_error_1, prob_error_2] = classify_FG_BG_MLE(img, seg_mask_gt, zigzag_pat_
    [img_height, img_width] = size(img);

    % number of training samples in Di for FG and BG
    num_TS_FG = size(TS_FG, 1);
    num_TS_BG = size(TS_BG, 1);

    % compute the covariance matrix of X|Y,T equal to the sample covariance
    TS_mu_FG = mean(TS_FG);
    TS_FG_bar = TS_FG - TS_mu_FG;
    TS_sigma_FG = (transpose(TS_FG_bar) * TS_FG_bar) / num_TS_FG;
    TS_sigma_FG_inv = inv(TS_sigma_FG);

    TS_mu_BG = mean(TS_BG);
    TS_BG_bar = TS_BG - TS_mu_BG;
    TS_sigma_BG = (transpose(TS_BG_bar) * TS_BG_bar) / num_TS_BG;
    TS_sigma_BG_inv = inv(TS_sigma_BG);

    % ML-estimate for class prior PY_FG and PY_BG
    num_TS = num_TS_FG + num_TS_BG;
    PY_FG = num_TS_FG / num_TS;   % P(Y = FG)
    PY_BG = num_TS_BG / num_TS;   % P(Y = BG)

    % classification of each pixel into FG and BG
    d = 64;                           % feature dimension
    block_dct_vec = zeros(d, 1);
    seg_mask_res = zeros(size(img));

    PYGX_FG_x = zeros(size(img));
    PYGX_BG_x = zeros(size(img));

    % pad test image with 7 layers to the right and bottom
    img_pad = img_padding(img);

    % for the MLE solution
    norm_const_FG = 1 / sqrt(power(2 * pi, d) * det(TS_sigma_FG));
    norm_const_BG = 1 / sqrt(power(2 * pi, d) * det(TS_sigma_BG));

    for i = 1:img_height
```

```
        for j = 1:img_width

            img_block_dct = dct2(img_pad(i:i+7,j:j+7));
            block_dct_vec(zigzag_pat_lin, 1) = img_block_dct(:);

            % MLE solution
            x_minus_mu_FG = block_dct_vec - transpose(TS_mu_FG);
            x_minus_mu_BG = block_dct_vec - transpose(TS_mu_BG);

            PXGY_x_FG = norm_const_FG * exp(-0.5 * transpose(x_minus_mu_FG) * TS_sigma_F
            PXGY_x_BG = norm_const_BG * exp(-0.5 * transpose(x_minus_mu_BG) * TS_sigma_B
            PX_x = PXGY_x_FG * PY_FG + PXGY_x_BG * PY_BG;

            PYGX_FG_x(i,j) = (PXGY_x_FG * PY_FG) / PX_x;
            PYGX_BG_x(i,j) = (PXGY_x_BG * PY_BG) / PX_x;

            if (PYGX_FG_x(i,j) > PYGX_BG_x(i,j))
                seg_mask_res(i,j) = 1;
            else
                seg_mask_res(i,j) = 0;
            end
        end
    end

    % compute prob of error for the MLE solution
    prob_error_1 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 1);
    prob_error_2 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 2);
    disp("Probability of Error (MLE Solution): "+prob_error_1);
end
```

2.4 Matlab Code - Function 'classify_FG_BG_MAP.m'

```
function [prob_error_1, prob_error_2] = classify_FG_BG_MAP(img, seg_mask_gt, zigzag_pat_
    [img_height, img_width] = size(img);

    % number of training samples in Di for FG and BG
    num_TS_FG = size(TS_FG, 1);
    num_TS_BG = size(TS_BG, 1);

    if strategy == 1
        % load prior1.mat
        prior1 = load("../data/Prior_1.mat");
        mu_prior_mu_FG = prior1.mu0_FG;
        mu_prior_mu_BG = prior1.mu0_BG;
        mu_prior_sigma_weights = prior1.W0;
```

```
    elseif strategy == 2
        % load prior2.mat
        prior2 = load("../data/Prior_2.mat");
        mu_prior_mu_FG = prior2.mu0_FG;
        mu_prior_mu_BG = prior2.mu0_BG;
        mu_prior_sigma_weights = prior2.W0;
    end

    mu_prior_sigma = alpha * diag(mu_prior_sigma_weights);   % same for FG and BG classe
    mu_prior_sigma_inv = inv(mu_prior_sigma);

    % compute the covariance matrix of X|Y,T equal to the sample covariance
    TS_mu_FG = mean(TS_FG);
    TS_FG_bar = TS_FG - TS_mu_FG;
    TS_sigma_FG = (transpose(TS_FG_bar) * TS_FG_bar) / num_TS_FG;
    TS_sigma_FG_inv = inv(TS_sigma_FG);

    TS_mu_BG = mean(TS_BG);
    TS_BG_bar = TS_BG - TS_mu_BG;
    TS_sigma_BG = (transpose(TS_BG_bar) * TS_BG_bar) / num_TS_BG;
    TS_sigma_BG_inv = inv(TS_sigma_BG);

    % compute posterior mean and covariance for FG class
    mu_post_sigma_FG = inv(num_TS_FG * TS_sigma_FG_inv + mu_prior_sigma_inv);
    mu_post_mu_FG = transpose((num_TS_FG * TS_mu_FG * TS_sigma_FG_inv + mu_prior_mu_FG *

    % compute posterior mean and covariance for BG class
    mu_post_sigma_BG = inv(num_TS_BG * TS_sigma_BG_inv + mu_prior_sigma_inv);
    mu_post_mu_BG = transpose((num_TS_BG * TS_mu_BG * TS_sigma_BG_inv + mu_prior_mu_BG *

    % compute parameters of predictive distribution for FG and BG class
    mu_pred_FG = mu_post_mu_FG;
    mu_pred_BG = mu_post_mu_BG;

    sigma_pred_FG = TS_sigma_FG;
%     sigma_pred_FG = mu_post_sigma_FG + TS_sigma_FG;
    sigma_pred_FG_inv = inv(sigma_pred_FG);

    sigma_pred_BG = TS_sigma_BG;
%     sigma_pred_BG = mu_post_sigma_BG + TS_sigma_BG;
    sigma_pred_BG_inv = inv(sigma_pred_BG);

    % ML-estimate for class prior PY_FG and PY_BG
    num_TS = num_TS_FG + num_TS_BG;
```

```
PY_FG = num_TS_FG / num_TS;   % P(Y = FG)
PY_BG = num_TS_BG / num_TS;   % P(Y = BG)

% classification of each pixel into FG and BG
d = 64;                           % feature dimension
block_dct_vec = zeros(d, 1);
seg_mask_res = zeros(size(img));

PYGX_FG_x = zeros(size(img));
PYGX_BG_x = zeros(size(img));

% pad test image with 7 layers to the right and bottom
img_pad = img_padding(img);

% predict cheetah image using BDR
norm_const_FG = 1 / sqrt(power(2 * pi, d) * det(sigma_pred_FG));
norm_const_BG = 1 / sqrt(power(2 * pi, d) * det(sigma_pred_BG));

for i = 1:img_height
    for j = 1:img_width

        img_block_dct = dct2(img_pad(i:i+7,j:j+7));
        block_dct_vec(zigzag_pat_lin, 1) = img_block_dct(:);

        x_minus_mu_FG = block_dct_vec - mu_pred_FG;
        x_minus_mu_BG = block_dct_vec - mu_pred_BG;

        PXGYT_x_FG = norm_const_FG * exp(-0.5 * transpose(x_minus_mu_FG) * sigma_pre
        PXGYT_x_BG = norm_const_BG * exp(-0.5 * transpose(x_minus_mu_BG) * sigma_pre
        PX_x = PXGYT_x_FG * PY_FG + PXGYT_x_BG * PY_BG;

        PYGX_FG_x(i,j) = (PXGYT_x_FG * PY_FG) / PX_x;
        PYGX_BG_x(i,j) = (PXGYT_x_BG * PY_BG) / PX_x;

        if (PYGX_FG_x(i,j) > PYGX_BG_x(i,j))
            seg_mask_res(i,j) = 1;
        else
            seg_mask_res(i,j) = 0;
        end
    end
end

% compute prob of error for the Bayesian solution
prob_error_1 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 1);
prob_error_2 = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, 2);
```

```
    disp("Probability of Error (MAP Solution): "+prob_error_1);

%   whos;
    f = figure();
    ax = gca;
    ax.FontSize = 16;
    f.WindowState = 'maximized';
    subplot(2,2,1);
    imshow(seg_mask_gt);
    title('Ground-Truth Mask')

    subplot(2,2,2);
    imshow(seg_mask_res);
    title('Result: 64-feature Mask')
    % imwrite(seg_mask_res_64, "../plots/seg_mask_res_64.png");

    subplot(2,2,3);
    imshow(img);
    title('Original Image')
    % imwrite(seg_mask_res_8, "../plots/seg_mask_res_best_8.png");

    subplot(2,2,4);
    imshow(img_pad);
    title('Padded Image')
    % imwrite(seg_mask_res_8, "../plots/seg_mask_res_best_8.png");

    close all;
end
```

2.5 Matlab Code - Function 'img_padding.m'

```
function [img_pad] = img_padding(img)
    [h, w] = size(img);
%     img_pad = zeros(h + 7, w + 7);
%     img_pad(4:h+3, 4:w+3) = img;
%
%     img_pad(1:3, 4:w+3) = img(1:3, :);
%     img_pad(h+4:end, 4:w+3) = img(h-3:h, :);
%     img_pad(4:h+3, 1:3) = img(:, 1:3);
%     img_pad(4:h+3, w+2:end) = img(:, w-7:w-2);
%     img_pad(1:3, 1:3) = img(1:3, 1:3);

    img_pad = padarray(img, [3 3], 'replicate', 'pre');
    img_pad = padarray(img_pad, [4 4], 'replicate', 'post');
```

```
%     img_pad = img(:,:);
%     img_pad(end+1:end+7,:) = img(end-7:end-1,:);
%     img_pad(1:end-7,end+1:end+7) = img(:,end-7:end-1);
%     img_pad(end-7:end,end-7:end) = img(end-7:end,end-7:end);
end
```

2.6 Matlab Code - Function 'compute_prob_error.m'

```
function [prob_error] = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, meth
    [h, w] = size(seg_mask_gt);

    if (method == 1)
        % Method-1 of computing probability of error
        num_FG_pixels = sum(seg_mask_gt, 'all');
        num_BG_pixels = h * w - num_FG_pixels;

        count_FG_pixels = 0;
        count_BG_pixels = 0;
        count_FG_error = 0;
        count_BG_error = 0;

        for i = 1:h
            for j = 1:w
                if (seg_mask_gt(i,j) == 1)
                    count_FG_pixels = count_FG_pixels + 1;
                    if (seg_mask_res(i,j) == 0)
                        count_FG_error = count_FG_error + 1;
                    end
                elseif (seg_mask_gt(i,j) == 0)
                    count_BG_pixels = count_BG_pixels + 1;
                    if (seg_mask_res(i,j) == 1)
                        count_BG_error = count_BG_error + 1;
                    end
                end
            end
        end

        assert(num_FG_pixels == count_FG_pixels);
        assert(num_BG_pixels == count_BG_pixels);

        prob_error = (count_FG_error / count_FG_pixels) * PY_FG + (count_BG_error / coun
    elseif (method == 2)
        % Method-2 of computing probability of error
        num_error_pixels = sum(abs(seg_mask_gt - seg_mask_res), 'all');
        num_pixels = h * w;
```

```
        prob_error = num_error_pixels / num_pixels;
    end
end
```