Parthasarathi Kumar
A59003519

Homework 3

Problem 4 (Quiz)

Introduction

In this assignment we do the segmentation task on 4 dataset, using 2 strategies for encoding the prior knowledge. There are 3 methods used for each of these, Maximum Likelihood(MLE), Bayesian Parameter Estimation (BPE) and Maximum a Posteriori(MAP).

In MLE, the class conditional density is modeled as a Gaussian(1) using the sample mean (2) and sample covaraince (3).

$$\underbrace{\boxed{\mathcal{P}_{X|T}\left(x|D\right) = \mathcal{N}\left(\hat{\mu}, \hat{\Sigma}\right)}}_{\text{MLE}} \tag{1}$$

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{1}^{n}\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)^T \tag{3}$$

In BPE, we assume a gaussian prior for the parameter μ as in (4)

$$\mathcal{P}_{\mu}\left(\mu\right) = \mathcal{N}\left(\mu_0, \Sigma_0\right) \tag{4}$$

where Σ_0 is parameterised by α as in (5).

$$\left(\Sigma_0\right) = \alpha w_{ii} \tag{5}$$

The resultant posterior mean is as shown by (6)

$$\mathcal{P}_{\mu|T}\left(\mu|D\right) = \mathcal{N}\left(\mu_n, \Sigma_n\right) \tag{6}$$

where

$$\mu_n = \Sigma_0\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\hat{\mu} + \frac{1}{n}\Sigma\left(\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\mu_0 \tag{7}$$

$$\Sigma_n = \Sigma_0\left(\Sigma_0 + \frac{1}{n}\Sigma\right)^{-1}\frac{1}{n}\Sigma \tag{8}$$

The class - conditional density given the posterior mean is -

$$\underbrace{\boxed{\mathcal{P}_{X|T}\left(x|D\right) = \mathcal{N}\left(\mu_n, \hat{\Sigma} + \Sigma_n\right)}}_{\text{BPE}} \tag{9}$$

where $\hat{\Sigma}$ is the sample variance as computed in (3).
In case of Maximum a Posteriori (MAP), we take the parameter (μ) with the maximum probability i.e. μ_n. The resulting class conditional density is -

$$\underbrace{\boxed{\mathcal{P}_{X|T}\left(x|D\right) = \mathcal{N}\left(\mu_n, \hat{\Sigma}\right)}}_{\text{MAP}} \tag{10}$$

(a) **PoE vs α plot for Dataset 1 & Strategy 1**
We observe in Figure 1, that the probability of error (PoE) increases with α implying that the estimate of the mean for lower values of α are more accurate. From (1) we can see that as α increases, the contribution of the sample mean (or μ_{ML}) increases. On the other hand, for lower α, the contribution from the prior is more. This indicates that the prior knowledge encoded in μ_0 is quite accurate.

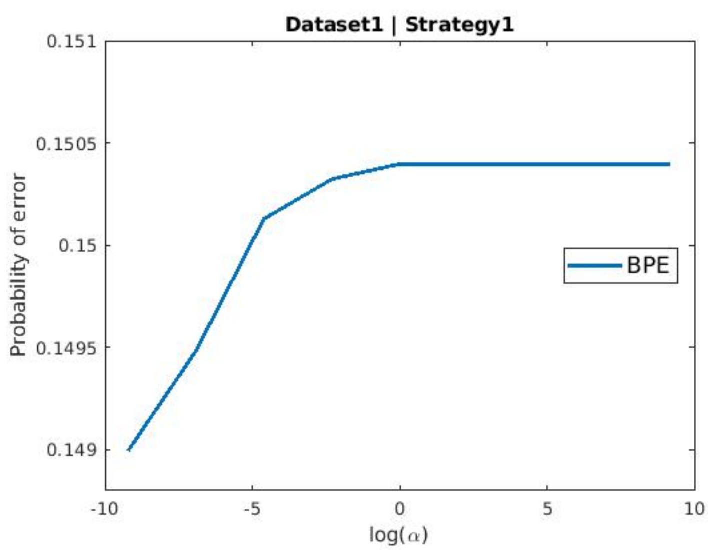

Figure 1: P(Error) vs α on data-set 1 & strategy 1 with different approaches

(b) **MLE vs BPE on Dataset 1**
The PoE for Maximum likelihood (MLE) technique is independent of α as we do not account for the prior and hence the straight line. As indicated in previous question, the contribution of μ_{MLE} increases with α, which in turn leads to the BPE tending to MLE as indicated in Figure 2. The relative performance between MLE and BPE is dependent on the priors

encoded. If the priors are informative, BPE performs better than MLE, otherwise it's vice versa as evident from Figure 5.

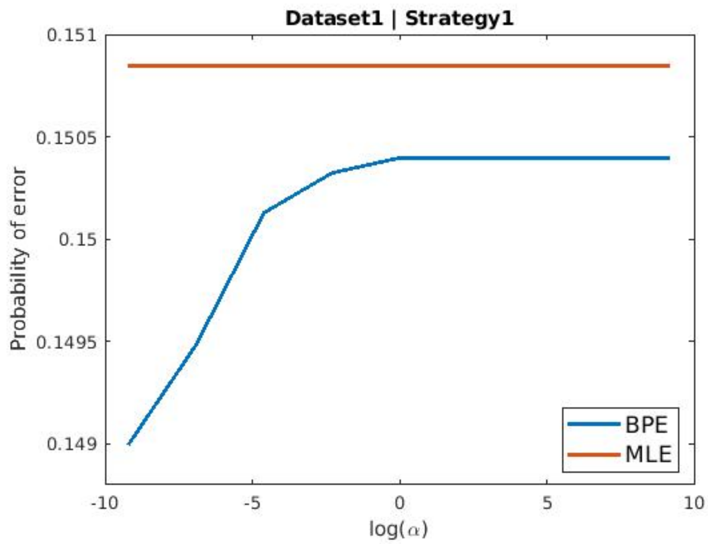

Figure 2: P(Error) vs α on data-set 1 & strategy 1 with BPE & MLE

(c) **MAP vs MLE & BPE for Dataset 1**
The MAP is an approximation technique over the BPE, where we take the most probable value of the parameter and hence performs worse than BPE which is a moer rigorous approach. The relative performace of MAP with MLE depends on the quality of the prior encoded. However, as α increases, the MAP tends to ML approach.

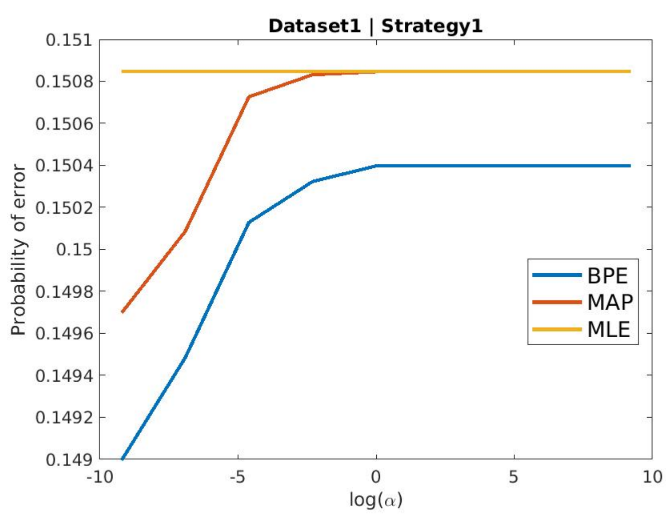

Figure 3: P(Error) vs α on data-set 1 & strategy 1 with BPE, MAP & MLE

(d) **Strategy 1 across Datesets 1,2,4 & 4 using the 3 techniques**
The trend described across the 3 methods for strategy 1 is similar for all the 4 data-sets. For data-set 2,3 & 4 both BPE & MAP give similar results.
However, there is a significant gap in case of data-set 1 (Figure 4) which has comparatively lesser data points. This can be accounted for the fact that BPE works well even when there is less training data. In such scenarios we rely on the prior knowledge. On the other hand, the performance of MLE improves as we have more data to model the distribution.
Strategy 2 across Datesets 1,2,4 & 4 using the 3 techniques
Trends across the 4 dataset for strategy 2 are similar to each other. Here too we observe a similar gap between MAP and BPE for dataset 1 due to the same reason stated above.

(e) **Strategy 1 vs Strategy 2**
The 2 strategies have different value for the mean of the gaussian prior. The first strategy captures the prior knowledge more accurately by allotting smaller μ_0 value for cheetah and larger μ_0 value for grass as is evident to the naked eye. On the other hand, strategy 2 assigns the same values across the 2 classes.
This leads to strategy 1 showing lower PoE for MAP or BPE than MLE for any dataset and α value. On the other hand for startegy 2 its the other way around where MLE gives a lower PoE.
The PoE for MLE is constant acorss α for the 2 strategies. As we increase the covaraince of the gaussian prior, BPE & MAP tend to MLE, especially MAP. This trend can be verified by comparing any 2 pair of plots across a dataset (Figure 5 & 6, for exaample).

Following are plots for P(error) vs α on each data-set and strategy.

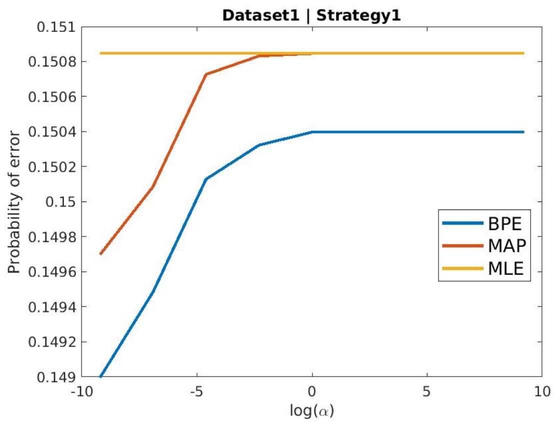

Figure 4: Strategy 1

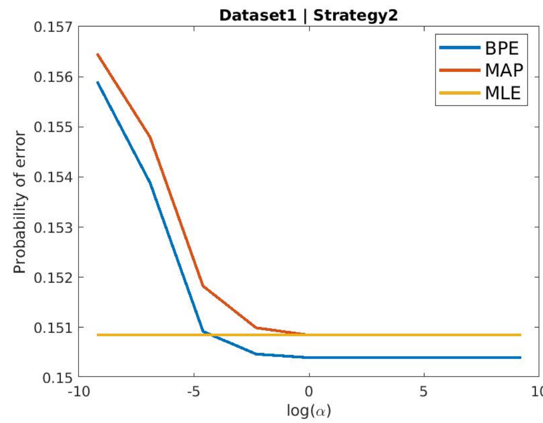

Figure 5: Strategy 2

Dataset 1

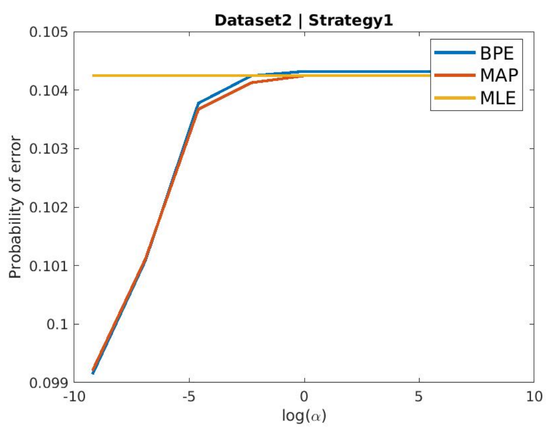

Figure 6: Strategy 1

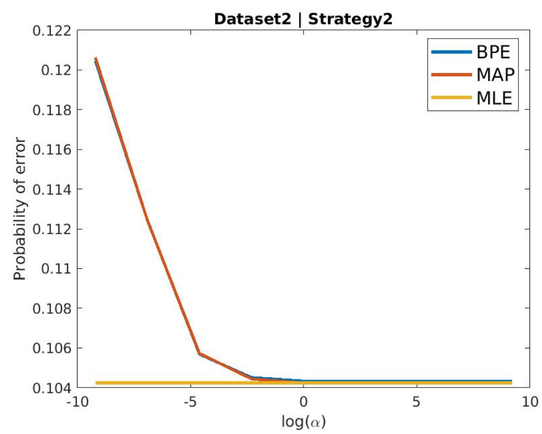

Figure 7: Strategy 2

Dataset 2

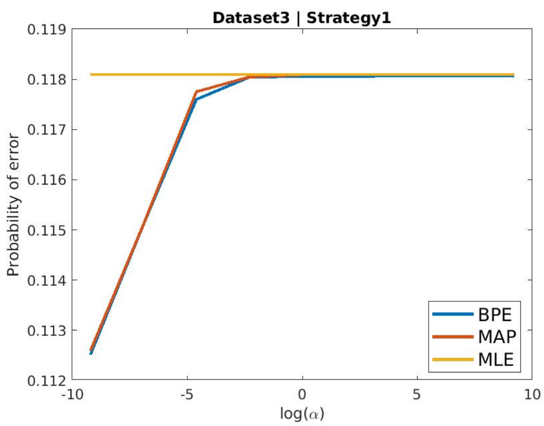

Figure 8: Strategy 1

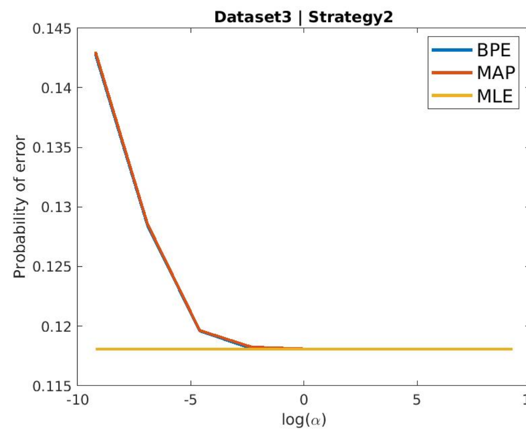

Figure 9: Strategy 2

Dataset 3

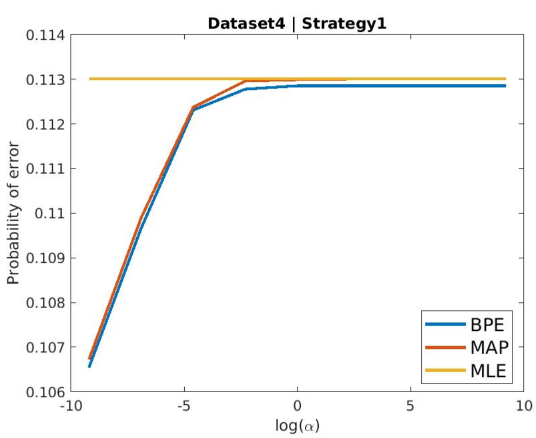

Figure 10: Strategy 1

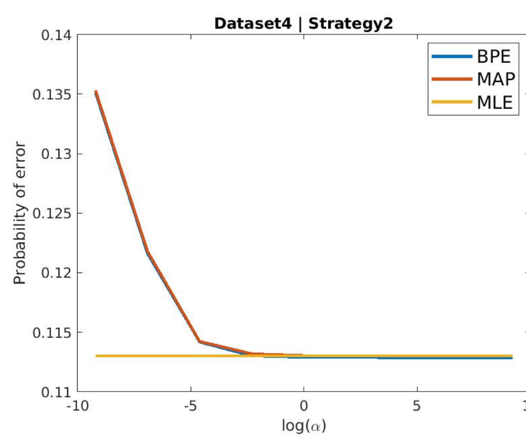

Figure 11: Strategy 2

Dataset 4

MATLAB Code - Main Experiment file

```
clc;
clear all;

% Driver to run the experiments across datasets, strategies and methods
    as
% in Assignment 3

% Handle folder to store results
if isfolder('errorResults')
    rmdir('errorResults', 's')
end
mkdir('errorResults');

% Load data
load('TrainingSamplesDCT_subsets_8.mat');

% Test image and gtruth
maskGT = imread('cheetah_mask.bmp');
I_cheetah = imread('cheetah.bmp');

% load alpha
load("Alpha.mat");

% Loop over strategy
for strategy = 1 : 2
    disp(strcat('Strategy -', int2str(strategy)));

    % Select strategy
    if (strategy == 1)
        load('Prior_1.mat');
    else
        load('Prior_2.mat');
    end

    % Loop over each dataset
    dataset = {'D1';'D2';'D3';'D4'};
    for d = 1 : length(dataset)
        disp(dataset{d});

        % Get data
        data_BG = eval(strcat(dataset{d},'_BG'));
```

```
        data_FG = eval(strcat(dataset{d},'_FG'));

        % To store probability of error
        pError = zeros(size(alpha));
        mask = {};
        % BPE for different alpha values
        for i = 1 : size(alpha,2)
            [mask{end+1}, pError(i)] = BPEEstimate(data_BG, data_FG,
               mu0_FG, mu0_BG, W0,...
                alpha(i), I_cheetah, maskGT);
        end

        name = getName(dataset(d), strategy, 'BPE');
        save(name,'pError');

        name = strcmp(name, 'mask');
        save(name, 'mask');

        % MLE for different alpha values
        mask = {};
        for i = 1 : size(alpha,2)
            [mask{end+1}, pError(i)] = MAPEstimate(data_BG, data_FG,
               mu0_FG, mu0_BG, W0,...
                alpha(i), I_cheetah, maskGT);
        end

        name = getName(dataset(d), strategy, 'MAP');
        save(name,'pError');

        name = strcmp(name, 'mask');
        save(name, 'mask');

        name = strcmp(name, 'mask');
        save(name, 'mask');

        % MLE for different alpha values
        mask = {};
        [mask{end+1}, pErrorMLE] = MLEEstimate(data_BG, data_FG,
           I_cheetah, maskGT);
        pError = repmat(pErrorMLE, size(pError));

        name = getName(dataset(d), strategy, 'MLE');
        save(name,'pError');
```

```
        name = strcmp(name, 'mask');
        save(name, 'mask');
    end
end

%% Helper Function
function name = getName(dataName, strategy, methods)
    % Returns an appropriate name for saving the error metric
    disp(methods);
    dataName = dataName{1};
    name = strcat(dataName, '_', methods, '_', int2str(strategy),'.mat'
        );
    name = fullfile(pwd, 'errorResults', name);
end
```

MATLAB Code - Bayesian Parameter Estimation

```
function [mask, pError] = BPEEstimate(data_BG, data_FG, mu_0_FG,
  mu_0_BG, W0,...
    alpha, I_cheetah, maskGT)

    % Function returns the probability of error on given test image and
    % gtruth mask.
    % The function expects the dataset of the 2 classes, and parameters
        of
    % the gaussian prior (mean).
    % Uses the Bayesian Parameter Estimation appraoch

    % Construct cov matrix of gaussian prior.
    cov_0_BG = alpha*diag(W0);
    cov_0_FG = alpha*diag(W0);

    % Calculate the predictive distribution parameters based on prior
      and data
    [mu_ccd_FG, cov_ccd_FG] = calcParamsPredictiveDist(data_FG, mu_0_FG
        , cov_0_FG);
    [mu_ccd_BG, cov_ccd_BG] = calcParamsPredictiveDist(data_BG, mu_0_BG
        , cov_0_BG);

    % Calculate prior probabilities
    p_BG = length(data_BG)/(length(data_BG) + length(data_FG));
    p_FG = 1 - p_BG;
```

```

    % Predict mask
    zigZagIdx = readmatrix('Zig-Zag Pattern.txt');
    mask = predictMask(cov_ccd_FG, cov_ccd_BG, mu_ccd_FG, mu_ccd_BG,...
        p_FG, p_BG, I_cheetah, zigZagIdx);

    % Calculate Error
    pError = calculateError(mask, maskGT, p_BG, p_FG);
end

%% Helper functions
function [mu_ccd, cov_ccd] = calcParamsPredictiveDist(data, mu_0, cov_0)
    % Given the prior and data, this function returns the parameters of the
    % predictive distribution
    cc_cov = cov(data);

    % Number of training data
    n = length(data);

    % Compute the sample means of the data.
    sample_mean = mean(data);

    % Calulate the mean and covaraince of the posteriror densities of the model
    % parameter (Mean of the Gaussian)
    mu_post  =  calculatePosteriorMean(cov_0, cc_cov, sample_mean, mu_0, n);
    cov_post =  calculatePosteriorCov(cov_0, cc_cov, n);

    % Calculate parameters of the predictive distribution
    mu_ccd = mu_post';
    cov_ccd = cc_cov + cov_post;
end

function mu_n = calculatePosteriorMean(sigma_0, sigma, mu_hat, mu_0, n)
    % Calculates posterior mean of model parameter (mean)
    % sigma_0 - Cov of prior
    % sigma - cov of ccd
    % mu_hat - sample mean
    % mu_0 - mean of prior
    % n - number of trainig data
```

```
    weighted_cov = pinv(sigma_0 + sigma/n);
    mu_n = sigma_0*weighted_cov*mu_hat' + (sigma*weighted_cov*mu_0')/n;
end

function sigma_n = calculatePosteriorCov(sigma_0, sigma, n)
    % Calculates posterior variance of model parameter (mean)
    % sigma_0 - cov of proir
    % sigma - cov of ccd
    % n - number of trainig data
    sigma_n = sigma_0*pinv((sigma_0 + sigma/n))*sigma/n;
end
```

MATLAB Code - Maximum a Posteriori Estimation

```
function [mask, pError] = MAPEstimate(data_BG, data_FG, mu_0_FG,
  mu_0_BG, W0,...
    alpha, I_cheetah, maskGT)

    % Function returns the probability of error on given test image and
    % gtruth mask.
    % The function expects the dataset of the 2 classes, and parameters
        of
    % the gaussian prior (mean).
    % Uses the Maximum a Posteriori appraoch

    % Construct cov matrix of gaussian prior.
    cov_0_BG = alpha*diag(W0);
    cov_0_FG = alpha*diag(W0);

    % Calculate the MAP estimate of the model parameter(mean).
    [mu_ccd_FG, cov_ccd_FG] = calcMAPEstimate(data_FG, mu_0_FG,
        cov_0_FG);
    [mu_ccd_BG, cov_ccd_BG] = calcMAPEstimate(data_BG, mu_0_BG,
        cov_0_BG);

    % Calculate prior probabilities
    p_BG = length(data_BG)/(length(data_BG) + length(data_FG));
    p_FG = 1 - p_BG;

    % Predict mask
    zigZagIdx = readmatrix('Zig-Zag Pattern.txt');
    mask = predictMask(cov_ccd_FG, cov_ccd_BG, mu_ccd_FG, mu_ccd_BG,...
        p_FG, p_BG, I_cheetah, zigZagIdx);
```

```
    % Calculate Error
    pError = calculateError(mask, maskGT, p_BG, p_FG);
end

%% Helper functions
function [mu_ccd, cov_ccd] = calcMAPEstimate(data, mu_0, cov_0)
    % Given the prior and data, this function returns the MAP estimate
        of
    % the parameter
    % calculate sample variance
    cc_cov = cov(data);

    % Number of training data
    n = length(data);

    % Compute the sample means of the data.
    sample_mean = mean(data);

    % Calulate the mean and covaraince of the posteriror densities of
        the model
    % parameter (Mean of the Gaussian)
    mu_post  =  calculatePosteriorMean(cov_0, cc_cov, sample_mean, mu_0
        , n);

    % Calculate parameters of the predictive distribution
    mu_ccd = mu_post';
    cov_ccd = cc_cov;
end

function mu_n = calculatePosteriorMean(sigma_0, sigma, mu_hat, mu_0, n)
    % Calculates posterior mean of model parameter (mean)
    % sigma_0 - Cov of prior
    % sigma - cov of ccd
    % mu_hat - sample mean
    % mu_0 - mean of prior
    % n - number of trainig data
    weighted_cov = pinv(sigma_0 + sigma/n);
    mu_n = sigma_0*weighted_cov*mu_hat' + (sigma*weighted_cov*mu_0')/n;
end
```

MATLAB Code - Maximum Likelihood Estimation

```
function [mask, pError] = MLEEstimate(data_BG, data_FG, I_cheetah,
    maskGT)

    % Function returns the probability of error on given test image and
    % gtruth mask.
    % The function expects the dataset of the 2 classes
    % Uses the Maximum likelihood approach

    % Calculate the MAP estimate of the model parameter(mean).
    [mu_ccd_FG, cov_ccd_FG] = calcMLEEstimate(data_FG);
    [mu_ccd_BG, cov_ccd_BG] = calcMLEEstimate(data_BG);

    % Calculate prior probabilities
    p_BG = length(data_BG)/(length(data_BG) + length(data_FG));
    p_FG = 1 - p_BG;

    % Predict mask
    zigZagIdx = readmatrix('Zig-Zag Pattern.txt');
    mask = predictMask(cov_ccd_FG, cov_ccd_BG, mu_ccd_FG, mu_ccd_BG,...
        p_FG, p_BG, I_cheetah, zigZagIdx);

    % Calculate Error
    pError = calculateError(mask, maskGT, p_BG, p_FG);
end

%% Helper functions
function [mu_ccd, cc_cov] = calcMLEEstimate(data)
    % Given the data, this function returns the MLE estimate of
    % the model
    cc_cov = cov(data);
    mu_ccd = mean(data);
end
```

MATLAB Code - Predict Mask

```
function mask = predictMask(cov_ccd_FG, cov_ccd_BG,...
    mu_ccd_FG, mu_ccd_BG, p_FG, p_BG, I_cheetah, zigZagIdx)

    % Function predicts the mask on test image based on parameters of
        the
    % posterior distribution and class priors
    alpha_FG = log(det(cov_ccd_FG))- 2*log(p_FG);
    alpha_BG = log(det(cov_ccd_BG))- 2*log(p_BG);
```

```
    % Predict mask for test image
    I_cheetah = im2double(I_cheetah);
    mask = zeros(size(I_cheetah));
    I_cheetah = padarray(I_cheetah,[7,7],'replicate','post');

    % Slide a 8X8 window over the image, calculate its DCT coeffecients. Select
    % the index of 2nd largest value as the feature to calculate posterior
    % probabilities.
    for i = 1 : 255
        for j = 1 : 270
            block = I_cheetah(i:i+7, j:j+7);
            dctF = dct2(block);
            fIdx(zigZagIdx(:)+1) = dctF(:);
            f = fIdx;

            dFG = (f - mu_ccd_FG)*inv(cov_ccd_FG)*(f - mu_ccd_FG)' + alpha_FG;
            dBG = (f - mu_ccd_BG)*inv(cov_ccd_BG)*(f - mu_ccd_BG)' + alpha_BG;
            if(dFG < dBG)
                mask(i,j) = 1;
            end
        end
    end
end
```

MATLAB Code - Calculate Error

```
function pError = calculateError(mask, gTruth, pB, pF)
    % Calculate probability of error given grounf truth mask, original
    % mask and class probabilities
    gTruth = im2double(gTruth);
    nCheetah = nnz(gTruth);
    nGrass = nnz(1 - gTruth);
    nMislabeledCheetah = nnz((mask-gTruth)>0);
    nMislabeledGrass = nnz((mask-gTruth)<0);
    pError = nMislabeledGrass/nGrass*pB + nMislabeledCheetah/nCheetah*pF;
end
```