Kushagra Agrawal
A53264533

Homework 3

Problem 4

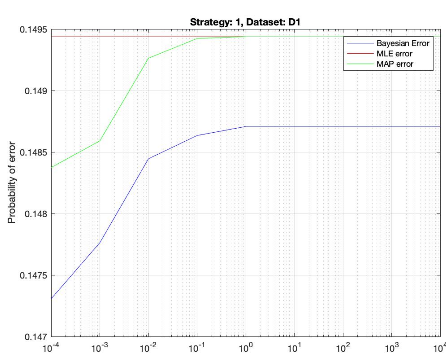
1. **Relative behavior of these three curves**
   Extending the derivations taught in class for univariate gaussian, the prior mean μ_1 and covariance Σ_1 can be calculated using the below equation:

$$\mu_1 = \Sigma_0(\Sigma_0 + \Sigma/N)^{-1}\mu_{ML} + \Sigma/N(\Sigma_0 + \Sigma/N)^{-1}\mu_0$$
$$\Sigma_1 = \Sigma_0(\Sigma_0 + \Sigma/N)^{-1}\Sigma/N$$

   Additionally, the desired predictive distribution can be derived using the below equation, using the fact that the product of two gaussian distributions is also a gaussian distribution:

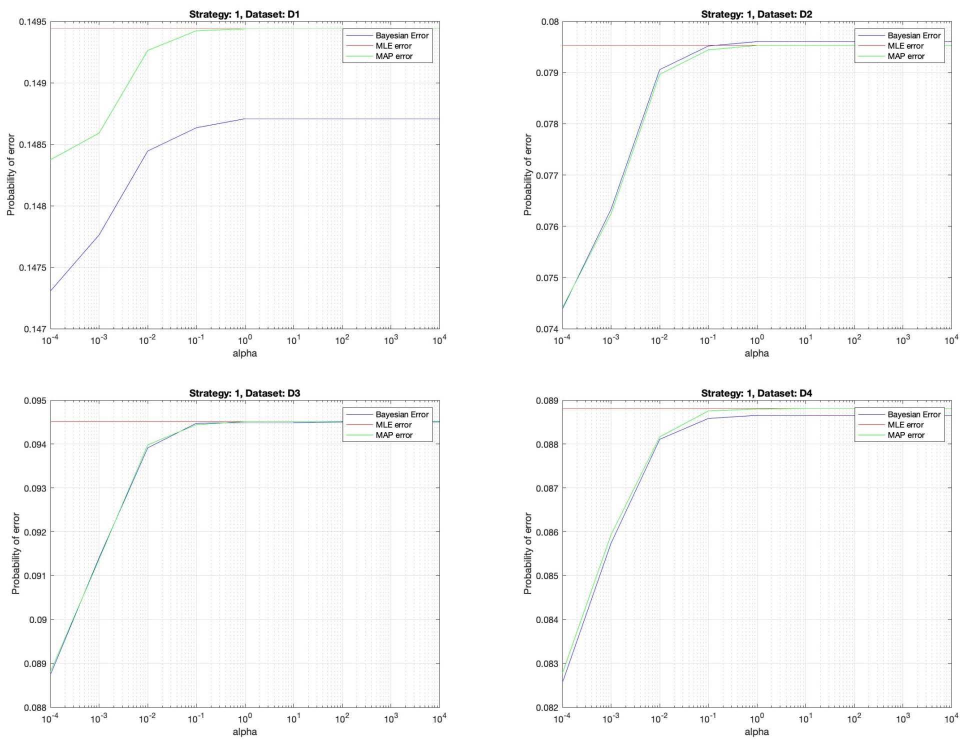
$$p(x|D) \sim N(\mu_1, \Sigma + \Sigma_1)$$

   Plotting the probability of error against increasing values of α, I observed that the probability of error increases for MAP and Bayesian estimation and remains constant for the maximum likelihood (ML) procedure. Since Maximum Likelihood parameters do not depend upon alpha, it is expectedly a horizontal line. For MAP and Bayesian estimation, since the prior covariance is defined as $(\Sigma_0)_{ii} = \alpha w_i$, the posterior mean, covariance clearly depend on it. When the value of alpha is small, the posterior mean μ_1 tends towards the prior probability μ_0. For increasing values of α, μ_1 tends closer to the ML mean μ_{ML} as given by the above equation. Additionally, since the covariance Σ_1 is the same for ML and MAP, we see the difference in probability of error being lesser when compared to ML and Bayesian estimation (for small values of alpha). Finally, since the probability of error is increasing with alpha, it also implies that the prior knowledge provided a better estimate of the parameters as compared to the maximum likelihood.

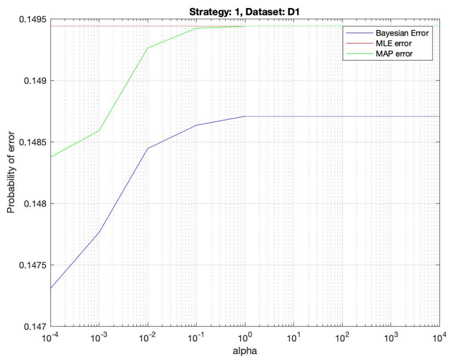


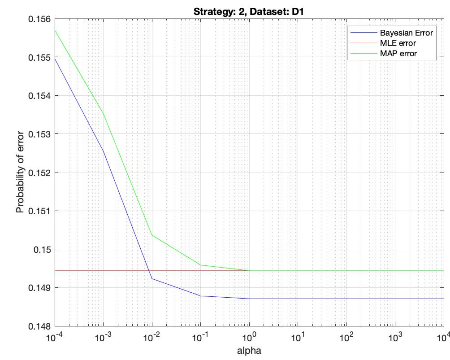
2. **How that behavior changes from dataset to dataset**
   From the above formula, it can be observed that the predictive distribution and MAP's

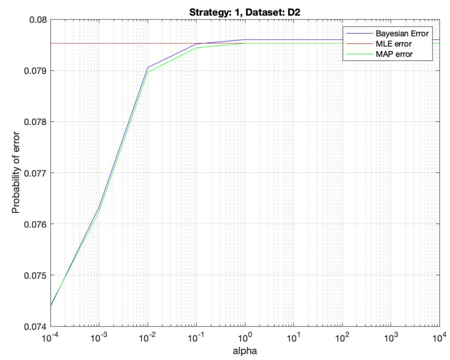
mean μ_1 is dependent on the number of samples N. As the number of samples increase, μ_1 will converge towards the ML solution μ_{ML} faster as compared to datasets with lower number of samples. Similarly, for covariance Σ_1, the value will converge to $\frac{\Sigma}{N}$ for MAP and predictive distribution.

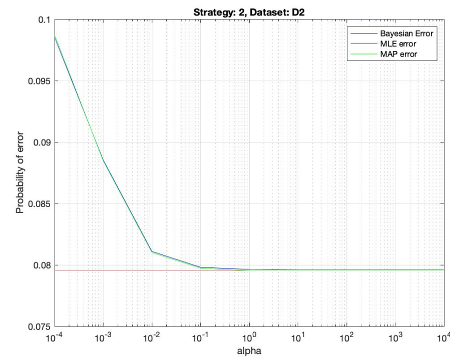


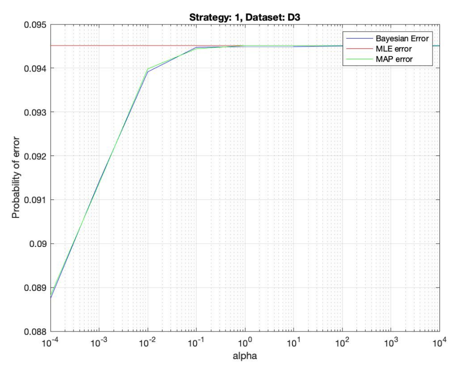
3. **How all of the above change when strategy 1 is replaced by strategy 2**
   In *strategy 1*, we assign a smaller weight for cheetah class ($\mu_0 = 1$) and larger for the grass class ($\mu_0 = 3$). This provides valuable prior information to the classifier in order to differentiate between the two classes. As mentioned before, for lower values of alpha, MAP and predictive distribution converge towards the Gaussian prior and towards ML as alpha increases, which can be observed in the graphs on the left. In contrast to this, *strategy 2* assigns equal weight to both the classes which does not provide the required prior knowledge to differentiate between the two classes. As alpha increases, the probability of error decreases and converges towards the ML.

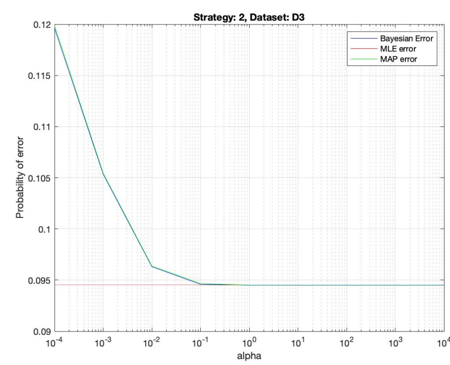

Strategy: 1, Dataset: D1
Bayesian Error
MLE error
MAP error
Probability of error
alpha

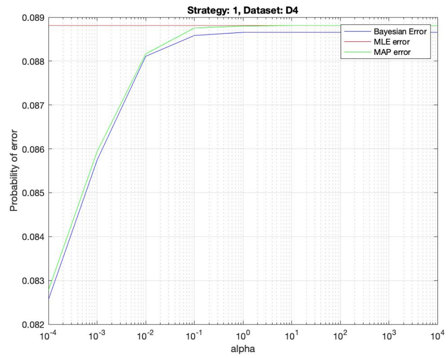

Strategy: 2, Dataset: D1
Bayesian Error
MLE error
MAP error
Probability of error
alpha

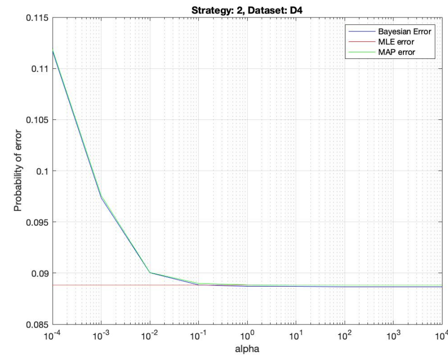

Strategy: 1, Dataset: D2
Bayesian Error
MLE error
MAP error
Probability of error
alpha


Strategy: 2, Dataset: D2
Bayesian Error
MLE error
MAP error
Probability of error
alpha


Strategy: 1, Dataset: D3
Bayesian Error
MLE error
MAP error
Probability of error
alpha


Strategy: 2, Dataset: D3
Bayesian Error
MLE error
MAP error
Probability of error
alpha





MATLAB code

```
clear;
clc;

trainingData = load('hw3Data/TrainingSamplesDCT_subsets_8.mat'
    );
alpha = load("hw3Data/Alpha.mat");
prior_1 = load("hw3Data/Prior_1.mat");
prior_2 = load("hw3Data/Prior_2.mat");

zigzagPattern = load('Zig-Zag Pattern.txt');
zigzagPattern = zigzagPattern + 1; % 1 indexing in MATLAB

for prior = 1:2
    for dataset_num = 1:4
        if(dataset_num == 1)
            foreground = trainingData.D1_FG;
            background = trainingData.D1_BG;
        elseif(dataset_num == 2)
            foreground = trainingData.D2_FG;
            background = trainingData.D2_BG;
        elseif(dataset_num == 3)
            foreground = trainingData.D3_FG;
            background = trainingData.D3_BG;
        elseif(dataset_num == 4)
            foreground = trainingData.D4_FG;
            background = trainingData.D4_BG;
        end

        if(prior == 1)
```

```
            W0 = prior_1.W0;
            mu0_FG = prior_1.mu0_FG;
            mu0_BG = prior_1.mu0_BG;
        else
            W0 = prior_2.W0;
            mu0_FG = prior_2.mu0_FG;
            mu0_BG = prior_2.mu0_BG;
        end
        [row_fg, col_fg] = size(foreground);
        [row_bg, col_bg] = size(background);

        priorYCheetah = row_fg / (row_bg + row_fg);
        priorYGrass = row_bg / (row_bg + row_fg);

        fgFeatureMean = sum(foreground) / row_fg; % MLE mean foreground
        bgFeatureMean = sum(background) / row_bg; % MLE mean background
        fgFeatureCov = cov(foreground); % MLE covariance foreground
        bgFeatureCov = cov(background); % MLE covariance background

        original_Image = imread('cheetah.bmp');
        pad_Image = padarray(original_Image, [7 7], 'replicate', 'pre');
        imageModified = im2double(pad_Image);
        [image_row, image_col] = size(imageModified);

        groundTruth = imread('cheetah_mask.bmp');
        groundTruthModified = im2double(groundTruth);

        groundTruthFGCount = 0;
        groundTruthBGCount = 0;
        for i = 1 : image_row - 7
            for j = 1 : image_col - 7
                if groundTruthModified(i, j) == 1
                    groundTruthFGCount = groundTruthFGCount + 1;
                else
                    groundTruthBGCount = groundTruthBGCount + 1;
```

```
                end
            end
        end

        errorBayesian = zeros(1, length(alpha.alpha));
        errorMLE = zeros(1, length(alpha.alpha));
        errorMAP = zeros(1, length(alpha.alpha));

        for alpha_val = 1:length(alpha.alpha)
            % calculation of P(theta | D)
            % posterior mean calculation
            sigma0FG = alpha.alpha(alpha_val) * diag(W0);
            alphaNFG_first = sigma0FG * inv(sigma0FG + (1/row_fg) * fgFeatureCov);
            alphaNFG_second = (1/row_fg) * fgFeatureCov * inv(sigma0FG + (1/row_fg) * fgFeatureCov);
            posteriorMeanFG_D1 = transpose(alphaNFG_first * fgFeatureMean' + alphaNFG_second * mu0_FG');

            sigma0BG = alpha.alpha(alpha_val) * diag(prior_1.W0);
            alphaNBG_first = sigma0BG * inv(sigma0BG + (1/row_bg) * bgFeatureCov);
            alphaNBG_second = (1/row_bg) * bgFeatureCov * inv(sigma0BG + (1/row_bg) * bgFeatureCov);
            posteriorMeanBG_D1 = transpose(alphaNBG_first * bgFeatureMean' + alphaNBG_second * mu0_BG');

            % posterior Covariance calculation
            posteriorCovFG_D1 = sigma0FG * inv(sigma0FG + (1/row_fg) * fgFeatureCov) * ((1/row_fg) * fgFeatureCov);
            posteriorCovBG_D1 = sigma0BG * inv(sigma0BG + (1/row_bg) * bgFeatureCov) * ((1/row_bg) * bgFeatureCov);

            % parameters of predictive distribution (mu_n, posteriorCov + priorCov)

            distributionCovFG = fgFeatureCov + posteriorCovFG_D1;
            distributionCovBG = bgFeatureCov +
```

```
        posteriorCovBG_D1;

        alphaFG = log(((2*pi)^64) * det(distributionCovFG)) - 2*log(priorYCheetah);
        alphaBG = log(((2*pi)^64) * det(distributionCovBG)) - 2*log(priorYGrass);

        alphaFG_MLE = log(((2*pi)^64) * det(fgFeatureCov)) - 2*log(priorYCheetah);
        alphaFG_MAP = alphaFG_MLE;
        alphaBG_MLE = log(((2*pi)^64) * det(bgFeatureCov)) - 2*log(priorYGrass);
        alphaBG_MAP = alphaBG_MLE;

        calculatedMask_Bayesian = zeros(image_row - 7, image_col - 7);
        calculatedMask_MLE = zeros(image_row - 7, image_col - 7);
        calculatedMask_MAP = zeros(image_row - 7, image_col - 7);

        for i = 1:image_row - 7
            for j = 1:image_col - 7
                block = imageModified(i:i+7, j: j+7);
                dctOutput = dct2(block);
                orderedDCTOutput(zigzagPattern(:)) = dctOutput(:);
                calculatedMask_Bayesian(i,j) = calculateMask(orderedDCTOutput, posteriorMeanFG_D1, posteriorMeanBG_D1, distributionCovFG, distributionCovBG, alphaFG, alphaBG);
                calculatedMask_MLE(i,j) = calculateMask(orderedDCTOutput, fgFeatureMean, bgFeatureMean, fgFeatureCov, bgFeatureCov, alphaFG_MLE, alphaBG_MLE);
                calculatedMask_MAP(i,j) = calculateMask(orderedDCTOutput, posteriorMeanFG_D1, posteriorMeanBG_D1, fgFeatureCov, bgFeatureCov, alphaFG_MAP, alphaBG_MAP);
```

```
                end
            end
            errorBayesian(alpha_val) = calculateErrorCount(
                groundTruthModified, calculatedMask_Bayesian,
                image_row-7, image_col-7, groundTruthFGCount,
                groundTruthBGCount, priorYCheetah, priorYGrass)
                ;
            errorMLE(alpha_val) = calculateErrorCount(
                groundTruthModified, calculatedMask_MLE,
                image_row-7, image_col-7, groundTruthFGCount,
                groundTruthBGCount, priorYCheetah, priorYGrass)
                ;
            errorMAP(alpha_val) = calculateErrorCount(
                groundTruthModified, calculatedMask_MAP,
                image_row-7, image_col-7, groundTruthFGCount,
                groundTruthBGCount, priorYCheetah, priorYGrass)
                ;
        end
        figure;
        semilogx(alpha.alpha, errorBayesian, '-b', alpha.alpha
            , errorMLE, '-r', alpha.alpha, errorMAP, '-g');
        grid on;
        titleText = ['Strategy: ', num2str(prior), ', Dataset:
            D', num2str(dataset_num)];
        title(titleText);
        legend('Bayesian Error', 'MLE error', 'MAP error');
        xlabel('alpha');
        ylabel('Probability of error');
    end
end

function probError = calculateErrorCount(groundTruthModified,
    mask, image_row, image_col, groundTruthFGCount,
    groundTruthBGCount, priorCheetah, priorGrass)
    errorFGCount = 0; % false negative
    errorBGCount = 0; % false positive
    for i = 1:image_row
        for j = 1:image_col
            if mask(i,j) ==0 && groundTruthModified(i, j) == 1
                errorFGCount = errorFGCount + 1;
            elseif mask(i,j) == 1 && groundTruthModified(i, j)
                == 0
```

```
                    errorBGCount = errorBGCount + 1;
                end
            end
        end

        fgError = errorFGCount / groundTruthFGCount;
        bgError = errorBGCount / groundTruthBGCount;

        probError = (fgError * priorCheetah) + (bgError *
            priorGrass);
end

function mask = calculateMask(dctOutput, meanFG, meanBG, fgCov
    , bgCov, alphaFG, alphaBG)
        mahalanobisFG = (dctOutput - meanFG) * inv(fgCov) *
            transpose(dctOutput - meanFG);
        mahalanobisBG = (dctOutput - meanBG) * inv(bgCov) *
            transpose(dctOutput - meanBG);
        if mahalanobisFG + alphaFG < mahalanobisBG + alphaBG
            mask = 1;
        else
            mask = 0;
        end
end
```