ECE 271A: Statistical Learning I - Quiz #3

Mazeyu Ji - A59023027

December 1, 2023

1. Basic principle

The following are the fundamental approaches used in statistical estimation and decision theory:

- **ML (Maximum Likelihood) Method**: The ML method seeks the parameter values that maximize the likelihood of the observed data. It is purely data-driven and does not incorporate prior information.

$$P_{x|T}(x|D) = \mathcal{G}(x, \mu_{ML}, \Sigma) \tag{1}$$

where μ_{ML} is the sample mean, and Σ is the sample covariance matrix calculated from the observed data.

- **MAP (Maximum A Posteriori) Method**: The MAP method incorporates prior knowledge through a prior distribution and updates this with the observed data to form a posterior distribution. The estimated parameters are those that maximize this posterior distribution.

$$P_{x|T}(x|D) = \mathcal{G}(x, \mu_n, \Sigma) \tag{2}$$

The posterior mean μ_n for MAP is calculated by blending the sample mean with the prior mean, weighted by the precision of the prior and the data:

$$\mu_n = \Sigma_0 \left(\Sigma_0 + \frac{1}{N}\Sigma\right)^{-1} \mu_{ML} + \left(\frac{1}{N}\Sigma \left(\Sigma_0 + \frac{1}{N}\Sigma\right)^{-1}\right) \mu_0 \tag{3}$$

where μ_0 and Σ_0 are the mean and covariance of the Gaussian prior, respectively.

- **Predictive Distribution**: The predictive distribution method goes a step further by considering not only the posterior distribution of the parameters but also their uncertainty. This leads to a distribution over new, unseen data points, integrating out the uncertainty.

$$P_{x|T}(x|D) = \mathcal{G}(x, \mu_n, \Sigma + \Sigma_n) \tag{4}$$

The adjusted covariance Σ_n for the predictive distribution takes into account the uncertainty about the parameter estimates:

$$\Sigma_n = \Sigma_0 \left(\Sigma_0 + \frac{1}{N} \Sigma \right)^{-1} \frac{1}{N} \Sigma \tag{5}$$

2. Running Results

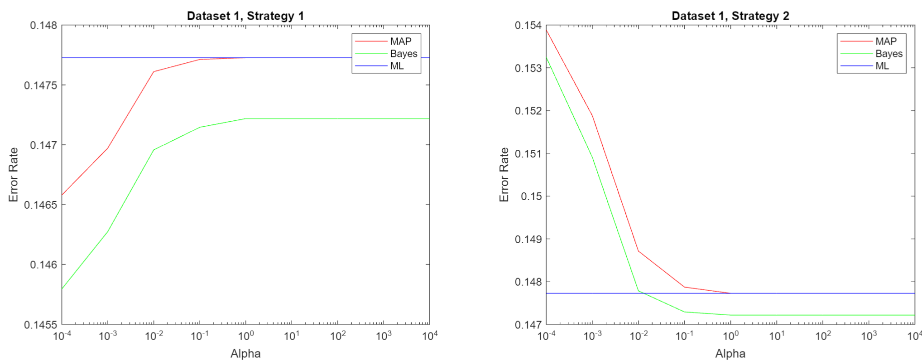

Figure 1: Comparison of error rates for different algorithms under dataset 1.

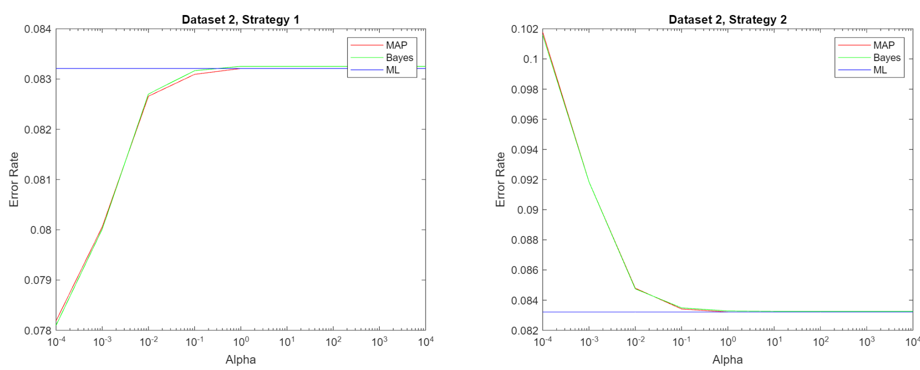

Figure 2: Comparison of error rates for different algorithms under dataset 2.

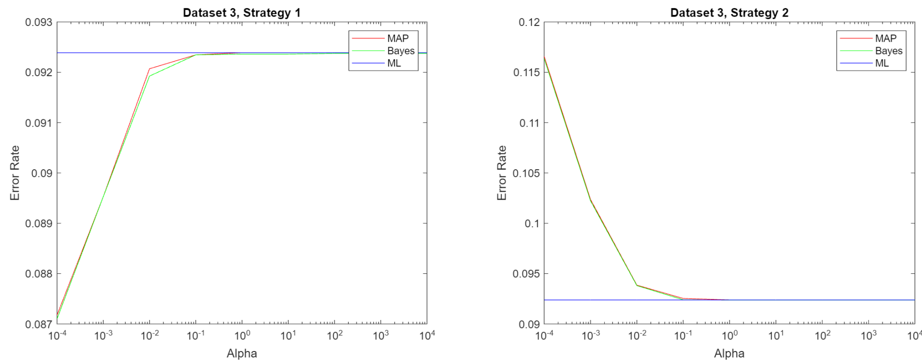

Figure 3: Comparison of error rates for different algorithms under dataset 3.

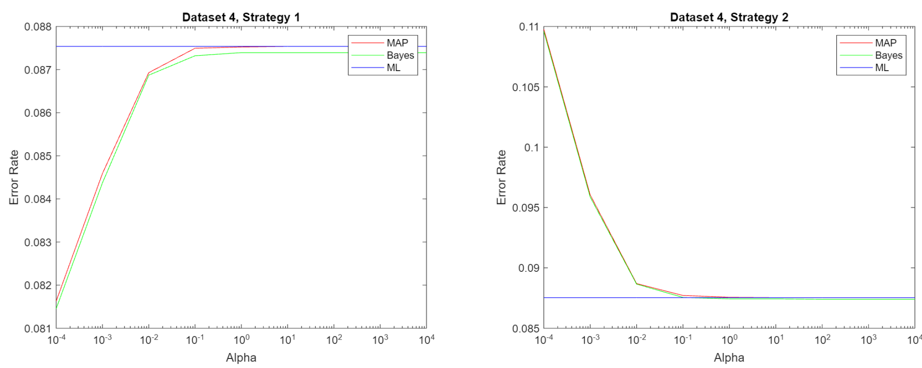

Figure 4: Comparison of error rates for different algorithms under dataset 4.

3. Results Analysis

(a) The Relative Behavior of These Three Curves

- **ML (Maximum Likelihood) Method:** The error rate (PoE) remains constant as the ML estimate does not depend on α, implying it does not incorporate prior information. It solely relies on the sample data to estimate the mean μ_{ML} and covariance Σ. In Strategy 1, the ML method underperforms in comparison to both MAP and the predictive distribution, whereas in Strategy 2, it outperforms them. This will be explained in the problem (c).

- **MAP (Maximum A Posteriori) Method:** In the MAP estimation, as α increases, the posterior mean μ_n approaches the ML estimate of the mean μ_{ML} due to the reduced influence of the prior on the mean. The covariance Σ remains unchanged in the MAP estimate, and therefore, as α becomes very large, the behavior of the MAP approaches that of the ML method, tending to rely entirely on the sample data.

- **Predictive Distribution:** In the predictive distribution, the posterior mean μ_n also approaches the ML estimate of the mean μ_{ML} with increasing α. However, its posterior covariance Σ_n incorporates a component of the prior covariance, resulting in a persistent difference from the ML method due to the term $\frac{1}{N} \times \Sigma$, even when α is large. This reflects a systematic difference of the predictive distribution from the ML method, which only relies on sample information, due to the integration of both prior and sample information.

(b) How That Behavior Changes From Dataset to Dataset

- For **smaller datasets** (such as D1), the limited sample size amplifies the influence of prior information on the posterior distribution. Consequently, the performance of the predictive distribution can significantly improve due to the integration of prior knowledge with sample data. Where sample information is insufficient for accurate parameter estimation, prior information provides a useful supplement.
- However, in **larger datasets** (such as D2, D3, and D4), the increase in sample size provides more substantial sample information, diminishing the relative impact of prior information. In these cases, the performance of the predictive distribution and MAP methods tend to converge, as the weight of prior information lessens and the influence of sample data becomes more pronounced.
- As the sample size grows further, the behaviors of the predictive distribution and MAP methods increasingly approximate the ML method, since all methods tend to rely more on sample data than prior knowledge. Nonetheless, since the predictive distribution method always incorporates the $\frac{1}{N} \times \Sigma$ term, it maintains a certain difference from the ML method even in large datasets, indicating that the predictive distribution method still considers a degree of prior knowledge even when data is abundant.

(c) How All of the Above Change When Strategy 1 Is Replaced by Strategy 2

- **Strategy 1** assumes different prior means for the two categories, reflecting differences in brightness between them. For the darker cheetah class, the prior mean is smaller; for the lighter grass class, the prior mean is larger.
- **Strategy 2** sets the same prior mean for both categories, which is based on half of the range of DCT coefficient amplitudes. This setting does not account for potential differences in brightness between categories.

Upon shifting to Strategy 2, we observe that:

- The performance of the predictive distribution and MAP methods may initially be worse than the ML method, especially at smaller values of α.

This is because the prior mean provided by Strategy 2 no longer differentiates between the two classes but instead offers a potentially inaccurate intermediate value.

- As α increases, the influence of the prior information diminishes, and the performance of MAP and predictive distribution methods gradually improves and begins to converge toward the performance of the ML method. This suggests that even less accurate prior information is overwhelmed by the information contained in the sample data when α is sufficiently large.
- In larger datasets, where sample data is abundant, the performance differences among the methods diminish. In this scenario, the impact of prior information—accurate or otherwise—becomes less significant, and the influence of sample data becomes more pronounced.

Thus, the change in strategy primarily affects performance at smaller values of α, where the impact of prior information on the posterior distribution is more significant. At larger values of α or in larger datasets, the specific choice of prior information has less impact on performance, and the methods tend to align, being mainly driven by the data.

Appendix

(a)solution.m

```
%% Calculate the error rates of different algorithms/strategies/datasets
% Load the alpha values and the datasets
load('Alpha.mat');
load('TrainingSamplesDCT_subsets_8.mat'); % Load the datasets

% Define the datasets for background (BG) and foreground (FG) for each D1, D2, D3, D4
datasets_BG = {D1_BG, D2_BG, D3_BG, D4_BG};
datasets_FG = {D1_FG, D2_FG, D3_FG, D4_FG};
prior_files = {'Prior_1.mat', 'Prior_2.mat'};

% Initialize an array to store error rates for 4 datasets, 2 strategies, each alpha value, and 3 error types
errors = zeros(4, 2, length(alpha), 3);

% Loop over each dataset
for i = 1:length(datasets_BG)
    D_BG = datasets_BG{i};
    D_FG = datasets_FG{i};

    % Loop over each strategy
    for j = 1:length(prior_files)
        prior_file = prior_files{j};
        k = 1;

        % Loop over each alpha value
        for a = alpha
            % Compute error rates for MAP, Bayes, and ML decision rules
            errors(i, j, k, 1) = MAP_BDR(D_BG, D_FG, a, prior_file);
            errors(i, j, k, 2) = Bayes_BDR(D_BG, D_FG, a, prior_file);
            k = k + 1;
        end
        errors(i, j, :, 3) = ML_BDR(D_BG, D_FG);
    end
end

%% Plotting the results
for i = 1:length(datasets_BG)
    figure('Position', [100, 100, 1200, 400]); % Adjust the size of the figure

```

```
    % Plot results for each strategy
    for j = 1:length(prior_files)
        subplot(1, 2, j);
        plot(alpha, squeeze(errors(i, j, :, 1)), 'r-'); hold
             on; % MAP error rate
        plot(alpha, squeeze(errors(i, j, :, 2)), 'g-'); hold
             on; % Bayes error rate
        plot(alpha, squeeze(errors(i, j, :, 3)), 'b-'); hold
             on; % ML error rate
        set(gca, 'XScale', 'log'); % Set x-axis to
            logarithmic scale
        xticks(alpha); % Set x-axis ticks to each alpha
            value
        title(sprintf('Dataset %d, Strategy %d', i, j)); %
            Title for each subplot
        xlabel('Alpha'); % X-axis label
        ylabel('Error Rate'); % Y-axis label
        legend('MAP', 'Bayes', 'ML'); % Legend
    end
end
```

(b)ML_BDR.m

```
function [error] = ML_BDR(trainBG, trainFG)
    load('TrainingSamplesDCT_subsets_8.mat');
    % Estimate the prior probabilities
    [rowFG, columnFG] = size(trainFG);
    [rowBG, columnBG] = size(trainBG);
    priorFG = rowFG / (rowBG + rowFG);
    priorBG = rowBG / (rowBG + rowFG);

    % Compute the parameters for ML
    meanFG = mean(trainFG);
    meanBG = mean(trainBG);
    covFG = cov(trainFG);
    covBG = cov(trainBG);

    % Read and preprocess the image
    img = imread('cheetah.bmp');
    imgDouble = im2double(img);
    [height, width] = size(imgDouble);

    % Get pattern index
    pattern = readmatrix('Zig-Zag Pattern.txt') + 1;

    % Calculate the threshold
    thresStar = priorBG / priorFG;

    % Loop over the image and make a decision
```

```
    maskRes = zeros(height, width);
    for i = 1:height-7
        for j = 1:width-7
            block = imgDouble(i:i+7, j:j+7);
            dctBlock = dct2(block);
            zigzag = zeros(1, 64);
            for m = 1:8
                for n = 1:8
                    zigzag(pattern(m,n)) = dctBlock(m,n);
                end
            end

            Px_yFG = my_mvnpdf(zigzag, meanFG, covFG);
            Px_yBG = my_mvnpdf(zigzag, meanBG, covBG);
            if Px_yFG / Px_yBG > thresStar
                maskRes(i, j) = 1;
            end

        end
    end

    % Compute error
    maskGT = imread('cheetah_mask.bmp');
    maskGT = im2double(maskGT);
    error = sum(sum(maskRes ~= maskGT)) / (height * width);
end
%% Define the function to calculate the PDF for Normal
    Distribution
function multi_pdf = my_mvnpdf(x, mu, Sigma)
    k = length(mu);
    multi_pdf = 1 / ((2 * pi)^(k/2) * sqrt(det(Sigma))) *
        exp(-0.5 * (x - mu) * inv(Sigma) * (x - mu)');
end
```

(c)MAP_BDR.m

```
function [error] = MAP_BDR(trainBG, trainFG, alpha, prior)
    load('TrainingSamplesDCT_subsets_8.mat');
    % Estimate the prior probabilities
    [rowFG, columnFG] = size(trainFG);
    [rowBG, columnBG] = size(trainBG);
    priorFG = rowFG / (rowBG + rowFG);
    priorBG = rowBG / (rowBG + rowFG);

    % Compute the parameters for MAP
    load(prior);
    covFG = cov(trainFG);
    covBG = cov(trainBG);

```

```
    covPrior = diag(alpha * W0);

    meanFG = mean(trainFG);
    meanFG = covPrior * inv(covPrior + covFG/size(trainFG,
        1)) * ...
         meanFG' + covFG/size(trainFG, 1) * inv(covPrior +
             covFG/size(trainFG, 1)) * mu0_FG';

    meanBG = mean(trainBG);
    meanBG = covPrior * inv(covPrior + covBG/size(trainBG,
        1)) * ...
         meanBG' + covBG/size(trainBG, 1) * inv(covPrior +
             covBG/size(trainBG, 1)) * mu0_BG';

    % Read and preprocess the image
    img = imread('cheetah.bmp');
    imgDouble = im2double(img);
    [height, width] = size(imgDouble);

    % Get pattern index
    pattern = readmatrix('Zig-Zag Pattern.txt') + 1;

    % Calculate the threshold
    thresStar = priorBG / priorFG;

    % Loop over the image and make a decision
    maskRes = zeros(height, width);
    for i = 1:height-7
        for j = 1:width-7
            block = imgDouble(i:i+7, j:j+7);
            dctBlock = dct2(block);
            zigzag = zeros(1, 64);
            for m = 1:8
                for n = 1:8
                    zigzag(pattern(m,n)) = dctBlock(m,n);
                end
            end

            Px_yFG = my_mvnpdf(zigzag, meanFG', covFG);
            Px_yBG = my_mvnpdf(zigzag, meanBG', covBG);
            if Px_yFG / Px_yBG > thresStar
                maskRes(i, j) = 1;
            end

        end
    end

    % Compute error
    maskGT = imread('cheetah_mask.bmp');
    maskGT = im2double(maskGT);
```

```
    error = sum(sum(maskRes ~= maskGT)) / (height * width);
end
%% Define the function to calculate the PDF for Normal Distribution
function multi_pdf = my_mvnpdf(x, mu, Sigma)
    k = length(mu);
    multi_pdf = 1 / ((2 * pi)^(k/2) * sqrt(det(Sigma))) * exp(-0.5 * (x - mu) * inv(Sigma) * (x - mu)');
end
```

(d)Bayes_BDR.m

```
function [error] = Bayes_BDR(trainBG, trainFG, alpha, prior)
    load('TrainingSamplesDCT_subsets_8.mat');
    % Estimate the prior probabilities
    [rowFG, columnFG] = size(trainFG);
    [rowBG, columnBG] = size(trainBG);
    priorFG = rowFG / (rowBG + rowFG);
    priorBG = rowBG / (rowBG + rowFG);

    % Compute the parameters for Bayes
    load(prior);
    covFG = cov(trainFG);
    covBG = cov(trainBG);

    covPrior = diag(alpha * W0);

    meanFG = mean(trainFG);
    meanFG = covPrior * inv(covPrior + covFG/size(trainFG, 1)) * ...
         meanFG' + covFG/size(trainFG, 1) * inv(covPrior + covFG/size(trainFG, 1)) * mu0_FG';
    covFG = covFG + covPrior * inv(covPrior + covFG/size(trainFG,1)) * covFG/size(trainFG,1);

    meanBG = mean(trainBG);
    meanBG = covPrior * inv(covPrior + covBG/size(trainBG, 1)) * ...
         meanBG' + covBG/size(trainBG, 1) * inv(covPrior + covBG/size(trainBG, 1)) * mu0_BG';
    covBG = covBG + covPrior * inv(covPrior + covBG/size(trainBG,1)) * covBG/size(trainBG,1);

    % Read and preprocess the image
    img = imread('cheetah.bmp');
    imgDouble = im2double(img);
    [height, width] = size(imgDouble);

    % Get pattern index
```

```
    pattern = readmatrix('Zig-Zag Pattern.txt') + 1;

    % Calculate the threshold
    thresStar = priorBG / priorFG;

    % Loop over the image and make a decision
    maskRes = zeros(height, width);
    for i = 1:height-7
        for j = 1:width-7
            block = imgDouble(i:i+7, j:j+7);
            dctBlock = dct2(block);
            zigzag = zeros(1, 64);
            for m = 1:8
                for n = 1:8
                    zigzag(pattern(m,n)) = dctBlock(m,n);
                end
            end

            Px_yFG = my_mvnpdf(zigzag, meanFG', covFG);
            Px_yBG = my_mvnpdf(zigzag, meanBG', covBG);
            if Px_yFG / Px_yBG > thresStar
                maskRes(i, j) = 1;
            end

        end
    end

    % Compute error
    maskGT = imread('cheetah_mask.bmp');
    maskGT = im2double(maskGT);
    error = sum(sum(maskRes ~= maskGT)) / (height * width);
end
%% Define the function to calculate the PDF for Normal
    Distribution
function multi_pdf = my_mvnpdf(x, mu, Sigma)
    k = length(mu);
    multi_pdf = 1 / ((2 * pi)^(k/2) * sqrt(det(Sigma))) *
        exp(-0.5 * (x - mu) * inv(Sigma) * (x - mu)');
end
```