ECE 271A HW3&4 Quiz

Chunlin Chen (PID: A59023021)

November 24, 2023

1 Bayesian Estimation

1.1 Covariance $\mathbf{\Sigma}$ of the class-conditional

According to our assumption, the covariance $\mathbf{\Sigma}$ of the class-conditional is just the sample covariance of the training data. We will use the following formula to compute $\mathbf{\Sigma}$.

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

1.2 Parameters of the Posterior Density $P_{\mu|\mathbf{T}}(\mu|\mathcal{D}_1)$

Given that $P_\mu(\mu) \sim \mathcal{N}(\mu_0, \mathbf{\Sigma}_0)$, using the result from DHS, we can know that the posterior density $P_{\mu|\mathbf{T}}(\mu|\mathcal{D}_1) \sim \mathcal{N}(\mu_1, \mathbf{\Sigma}_1)$, and we can compute its parameters using the following formula:

$$\mu_1 = \mathbf{\Sigma}_0\left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma}\right)^{-1}\hat{\mu}_1 + \frac{1}{N}\mathbf{\Sigma}\left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma}\right)^{-1}\mu_0 \tag{2}$$

$$\mathbf{\Sigma}_1 = \mathbf{\Sigma}_0\left(\mathbf{\Sigma}_0 + \frac{1}{N}\mathbf{\Sigma}\right)^{-1}\frac{1}{N}\mathbf{\Sigma} \tag{3}$$

where $\hat{\mu}_1$ is the sample mean of the training data.

1.3 Parameters of the Predictive Distribution $P_{\mathbf{x}|\mathbf{T}}(\mathbf{x}|\mathcal{D}_1)$

We can also use the results from DHS, which tells us that $P_{\mathbf{x}|\mathbf{T}}(\mathbf{x}|\mathcal{D}_1) \sim \mathcal{N}(\mu_1, \mathbf{\Sigma}+\mathbf{\Sigma}_1)$.

1.4 Classification

For the class priors, we will just use the ML estimates, i.e. the fraction of the number of data for each class. Then, we plug the above results into the Bayesian Decision Rule to accomplish the classification task. The result is shown in Figure 1, we can see that as α increases, the probability of error increases, but the curve converges when α is greater than or equal to 0.1. As we know from the results of posterior density parameters above, a larger α means a decreasing confidence in the priors, which indicates that for Dataset 1 and Strategy 1, if we have a strong confidence in priors, the classification result will be better.

2 Maximum Likelihood Estimation

The MLE method just assume that $P_{\mathbf{x}|\mathbf{T}}(\mathbf{x}|\mathcal{D}_1) \sim \mathcal{N}(\hat{\mu}_1, \mathbf{\Sigma})$, where $\hat{\mu}_1$ is the sample mean of the training data, and then plug the estimates into Bayesian Decision Rule. The result is shown in Figure 1. Since the MLE method only relies on the sample mean and covariance, its result won't be influenced by the change of the α value. Moreover, the MLE method performs worse than the predictive, since the volume of this dataset is not very large, relying on the sample only without considering the priors could lead to bad results.

3 Bayesian Estimation with MAP Approximation

The MAP approximation for the Bayesian estimation choose θ that maximizes the posterior density $P_{\Theta|T}(\theta|D)$. As we have discussed in 1.2, $P_{\mu|\mathbf{T}}(\mu|\mathcal{D}_1) \sim \mathcal{N}(\mu_1, \mathbf{\Sigma}_1)$, therefore $\mu_{MAP} = \underset{\mu}{argmax}\, P_{\mu|\mathbf{T}}(\mu|\mathcal{D}_1) = \mu_1$, leading to $P_{\mathbf{x}|\mathbf{T}}(\mathbf{x}|\mathcal{D}_1) = P_{\mathbf{x}|\mu}(\mathbf{x}|\mu_{MAP}) \sim \mathcal{N}(\mu_1, \mathbf{\Sigma})$. Then we can plug the estimates into Bayesian Decision Rule. The result is shown in Figure 1, the curve also increases as α increases. As we see, when α is greater than 1, i.e. we have little confidence in our priors, the curve converges to the MLE method, this is because when we only take the sample data into consideration, the MAP method is approximately equivalent to the MLE method.

4 Results Analysis

We repeat the above procedure for different datasets under different strategies for the selection of the prior parameters, and the results are shown in Figure 1.

4.1 Relative Behavior of the three Curves

As we can see, the curves of the predictive method and MAP are influenced by the value of α and will converge as α reaches a certain value. MAP usually converges to ML since they are intrinsically equivalent when we have little confidence in priors and rely on the sample data dominantly.

4.2 Change of Behaviors across Datasets

As the scale of datasets increases, we can find that the overall probability of error shows a decreasing trend, though this trend is not strictly reflected between Dataset 2, 3 and 4, and they all outperform Dataset 1. This indicates that in general, a relatively larger scale of training data will lead to better classification results than a small dataset, but it does not mean that the larger, the better, considering other problems such as overfitting. In addition, we can see that the predictive method and MAP all converge to ML in the larger datasets, which is because when the number of samples is large enough, even the Bayesian estimation is dominantly decided by the sample data, therefore the differences between the three methods are negligible.

4.3 Change of Behaviors under Different Strategies

Obviously, the curves under different strategies display totally opposite trends. As the value of α increases, i.e. our confidence in priors decreases, the curves predictive method and MAP increase under Strategy 1 and decrease while under Strategy 2. And we can see that the probabilities of error is overall larger under Strategy 2. The only difference of the two strategies lies in their prior assumptions for μ_0, which shows that Strategy 1 provides an excellent prior, i.e. $\mu_0 = 1$ for the *cheetah* class and $\mu_0 = 3$ for the *grass* class, which we can have strong confidence in. While Strategy 2 provides a poor non-informative prior which fails to distinguish between the two classes, i.e. $\mu_0 = 2$ for both classes, leading to bad classification results.

The above analysis inspires us that when we have strong priors we should have more confidence in the priors, while we have poor priors we should have less confidence and focus more on the sample data. And it is always better to have a relatively large dataset to extract more features from than a dataset containing sparse data samples. For small datasets like Dataset 1, the predictive method will perform better than MAP and ML as long as we choose a reasonable balance between the confidence in priors and sample data.

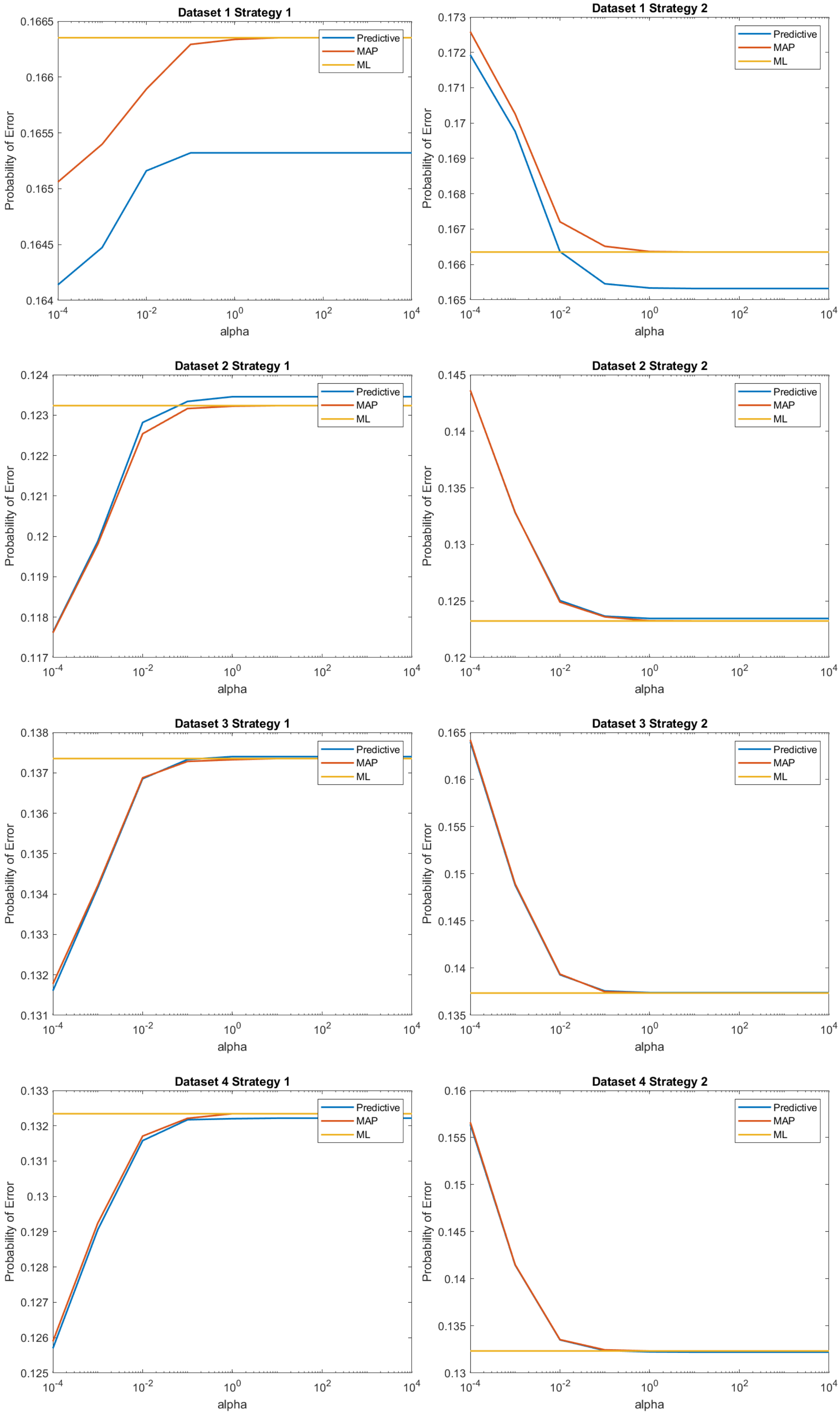

Figure 1: Curves of the Probability of Error on Different Datasets under Different Strategies

5 Code

```
load("HW3\hw3Data\TrainingSamplesDCT_subsets_8.mat")
load("HW3\hw3Data\Alpha.mat")

ZigZagPattern = readmatrix("HW1\Zig-Zag Pattern.txt");
ZigZagPattern = ZigZagPattern + 1;
ZigZagPattern = int8(ZigZagPattern);

img = imread("HW1\cheetah.bmp");
img = im2double(img);

% Zero Padding
left = zeros(255, 4);
right = zeros(255, 3);
up = zeros(4, 277);
bottom = zeros(3, 277);
img_pad = [up; [left img right]; bottom];

% DCT
img_dct = dct_8(img, img_pad);

% ZigZag Scan
img_scan = blockproc(img_dct, [8 8], @(block_struct)
    ZigZagScan(block_struct.data, ZigZagPattern));

ground_truth = imread("HW1\cheetah_mask.bmp");
ground_truth = im2double(ground_truth);

D_BG_all = [D1_BG; D2_BG; D3_BG; D4_BG];
D_BG_index = [0, size(D1_BG, 1), size(D2_BG, 1), size(
    D3_BG, 1), size(D4_BG, 1)];
D_FG_all = [D1_FG; D2_FG; D3_FG; D4_FG];
D_FG_index = [0, size(D1_FG, 1), size(D2_FG, 1), size(
    D3_FG, 1), size(D4_FG, 1)];

p_bbdr = zeros(2, 4, size(alpha, 2)); % Bayesian BDR
```

```
p_bmapbdr = zeros(2, 4, size(alpha, 2)); % Bayesian MAP BDR
p_mlbdr = zeros(2, 4, size(alpha, 2)); % ML BDR

for s = 1:2 % for each strategy
    if s == 1
        load("HW3\hw3Data\Prior_1.mat")
    end
    if s == 2
        load("HW3\hw3Data\Prior_2.mat")
    end

    for d = 1:4 % for each dataset
        s_bg = 1;
        s_fg = 1;
        for i = 1:d
            s_bg = s_bg + D_BG_index(1, i);
            s_fg = s_fg + D_FG_index(1, i);
        end
        e_bg = s_bg + D_BG_index(1, d + 1) - 1;
        e_fg = s_fg + D_FG_index(1, d + 1) - 1;
        D_BG = D_BG_all(s_bg:e_bg, :);
        D_FG = D_FG_all(s_fg:e_fg, :);

        % Compute the priors
        prob_bg = size(D_BG, 1) / (size(D_BG, 1) + size(D_FG, 1))
        prob_fg = size(D_FG, 1) / (size(D_BG, 1) + size(D_FG, 1))

        % Compute the covariance of the class-conditional - D1
        mean_bg = mean(D_BG, 1);
        cov_bg = 0;
        for i = 1:size(D_BG, 1)
            cov_bg = cov_bg + (D_BG(i, :) - mean_bg).'*(D_BG(i, :) - mean_bg)./size(D_BG, 1);
        end
```

```
        mean_fg = mean(D_FG, 1);
        cov_fg = 0;
        for i = 1:size(D_FG, 1)
            cov_fg = cov_fg + (D_FG(i, :) - mean_fg).'*(
               D_FG(i, :) - mean_fg)./size(D_FG, 1);
        end

        for j = 1:size(alpha, 2)
            % Compute the posterior mean and covariance
            mu_0_bg = mu0_BG;
            mu_0_fg = mu0_FG;
            Sigma_0 = zeros(64, 64);
            for i = 1:64
                Sigma_0(i, i) = alpha(1, j) * W0(1, i);
            end

            n_bg = size(D_BG, 1);
            n_fg = size(D_FG, 1);
            mu_n_bg = (Sigma_0*inv(Sigma_0+cov_bg/n_bg)*
               mean_bg.'+cov_bg*inv(Sigma_0+cov_bg/n_bg)*
               mu_0_bg.'/n_bg).';
            sigma_n_bg = Sigma_0*inv(Sigma_0+cov_bg/n_bg)/
               n_bg*cov_bg;
            mu_n_fg = (Sigma_0*inv(Sigma_0+cov_fg/n_fg)*
               mean_fg.'+cov_fg*inv(Sigma_0+cov_fg/n_fg)*
               mu_0_fg.'/n_fg).';
            sigma_n_fg = Sigma_0*inv(Sigma_0+cov_fg/n_fg)/
               n_fg*cov_fg;

            %% Bayesian BDR
            mask_bbdr = blockproc(img_scan, [1, 64], @(
               block_struct) BDR(block_struct.data,
               mu_n_bg, mu_n_fg, cov_bg+sigma_n_bg, cov_fg
               +sigma_n_fg, prob_bg, prob_fg));
            p_bbdr(s, d, j) = P_Error(ground_truth,
               mask_bbdr, prob_bg, prob_fg);

```

```
            %% Bayes MAP-BDR
            mask_bmapbdr = blockproc(img_scan, [1, 64], @(block_struct) BDR(block_struct.data, mu_n_bg, mu_n_fg, cov_bg, cov_fg, prob_bg, prob_fg));
            p_bmapbdr(s, d, j) = P_Error(ground_truth, mask_bmapbdr, prob_bg, prob_fg);
        end

        %% ML-BDR
        mask_mlbdr = blockproc(img_scan, [1, 64], @(block_struct) BDR(block_struct.data, mean_bg, mean_fg, cov_bg, cov_fg, prob_bg, prob_fg));
        p_mlbdr(s, d, :) = P_Error(ground_truth, mask_mlbdr, prob_bg, prob_fg);

        % Plot PoE
        f = figure((s-1)*4+d);
        slg = semilogx(alpha, reshape(p_bbdr(s, d, :), [1, 9]), alpha, reshape(p_bmapbdr(s, d, :), [1, 9]), alpha, reshape(p_mlbdr(s, d, :), [1, 9]));
        title(join(["Dataset", int2str(d), "Strategy", int2str(s)]))
        xlabel("alpha")
        ylabel("Probability of Error")
        legend('Predictive', 'MAP', 'ML')
        slg(1).LineWidth = 1.5;
        slg(2).LineWidth = 1.5;
        slg(3).LineWidth = 1.5;
        exportgraphics(f, append('d', int2str(d), 's', int2str(s), '.png'))
    end
end

function vector = ZigZagScan(matrix, pattern)
    vector = zeros(1, size(matrix, 1) * size(matrix, 2));
    for i = 1:size(matrix, 1)
        for j = 1:size(matrix, 2)
```

```
            position = pattern(i, j);
            vector(1, position) = matrix(i, j);
        end
    end
end

function mask = BDR(feature, mu_bg, mu_fg, sigma_bg,
   sigma_fg, P_bg, P_fg)
    if (feature-mu_bg)*inv(sigma_bg)*(feature-mu_bg).'+log
       ((2*pi).^64*det(sigma_bg))-2*log(P_bg)...
            < (feature-mu_fg)*inv(sigma_fg)*(feature-mu_fg
               ).'+log((2*pi).^64*det(sigma_fg))-2*log(
               P_fg)
        mask = 0;
    else
        mask = 1;
    end
end

function dct = dct_8(img, img_pad)
    dct = zeros(size(img, 1) * 8, size(img, 2) * 8);
    for i = 1:size(img, 1)
        for j = 1:size(img, 2)
            dct((8*i-7):(8*i), (8*j-7):(8*j)) = dct2(
               img_pad(i:i+7, j:j+7));
        end
    end
end

function p = P_Error(gt, mask, prob_bg, prob_fg)
    gt = int8(gt);
    mask = int8(mask);
    diff = gt - mask;
    detect = 1 - sum(sum(diff==1))/sum(sum(gt==1));
    fAlarm = sum(sum(diff==-1))/sum(sum(gt==0));
    p = fAlarm * prob_bg + (1 - detect) * prob_fg;
end
```