Student Name: Lin-Hsi (Ethen) Tsao
Student PID: A59015911

Statistical Learning 1 (ECE 271A)

HW3, 4

(a) **Consider the training set D_1 and strategy 1 (we will use this strategy until d)). For each class, compute the covariance $\sum$ of the class-conditional, and the posterior mean μ_1, and covariance $\sum_1$ of**

$$P_{\mu|T}(\mu|D_1) = G(\mu, \mu_1, \sum_1).$$

Next, compute the parameters of the predictive distribution

$$P_{x|T}(x|D_1)$$

for each of the classes. Then, using ML estimates for the class priors, plug into the Bayesian decision rule, classify the cheetah image and measure the probability of error. All of the parameters above are functions of α. Repeat the procedure for the values of α given in the file Alpha.mat. Plot the curve of the probability of error as a function of α. Can you explain the results?

First, for each class, we can easily compute the $\sum$ by

$$\sum = \frac{1}{N}\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i)(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i)^T$$

Then, we want to compute the posterior mean μ_1, and covariance $\sum_1$ of $P_{\mu|T(\mu|D_1)} = G(\mu, \mu_1, \sum_1)$ given $P_{x|\mu,\sum} = G(x, \mu, \sum)$ and $P_\mu(\mu) = G(x, \mu_0, \sum_0)$.

By Bayesian Solution, we can know that

$$P_{\mu|T}(\mu|D) \propto \prod_i P_{X|\mu}(x_i|\mu)P_\mu(\mu)$$

By taking the log of $P_{\mu|T}(\mu|D)$, we can further compute

$log(P_{\mu|T}(\mu|D)) = \frac{1}{2}\mu^T(N\sum^{-1} + \sum_0^{-1})\mu + \mu^T(\sum_0^{-1}\mu_0 + \sum^{-1}\sum_{i=1} Nx_i) + C$, where C is a constant

We know that the product of Gaussian PDF is still a Gaussain PDF. Therefore, we rewrite the $log(P_{\mu|T}(\mu|D))$:

$$log(P_{\mu|T}(\mu|D)) = \frac{1}{2}(\mu - A)^T(N\sum^{-1} + \sum_0^{-1})(\mu - A)$$

$$A = (N\sum^{-1} + \sum_0^{-1})^{-1}(\sum_0^{-1}\mu_0 + \sum^{-1}\sum_{i=1}^{N} x_i)$$

We know that $P_{\mu|T}(\mu|D_1) = G(\mu, \mu_1, \sum_1)$. Therefore we get

$$\mu_1 = A$$

$$\sum_1 = (N\sum^{-1} + \sum_0^{-1})^{-1}$$

By further calculation, we can compute that

$$\mu_1 = \sum_0(\sum_0 + \frac{1}{N}\sum)^{-1}(\frac{1}{N}\sum_{i=1}^{N} x_i) + \frac{1}{N}\sum(\sum_0 + \frac{1}{N}\sum)^{-1}\mu_0$$

$$\sum_1 = \sum_0(\sum_0 + \frac{1}{N}\sum^{-1})\frac{1}{N}\sum$$

Using the equation above, we can compute μ_1 and $\sum_1$. Next,

$$P_{x|T}(x|D_1) = \int_\mu (\frac{1}{\sqrt{2\pi\sigma^2}} \exp\frac{-(x-\mu)^2}{2\sigma^2})\,(\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\frac{-(x-\mu_1)^2}{2\sigma_1^2})d\mu$$

$$= G(x, 0, \sum) * G(x, \mu_1, \sum_1)$$

$$= G(x, \mu_1, \sum + \sum_1)$$

After we have the posterior, we apply the BDR to predict the label given a pixel. Therefore, the predicted label of a pixel is the foreground if

$$log(P_{X|T}(x|D_{FG})\,P_T(D_{FG})) \geq log(P_{X|T}(x|D_{BG})\,P_T(D_{BG}))$$

Otherwise, the predicted label is background.

The curve of the PD is shown in Figure 1, Strategy 1 Dataset 1. Now, we only focus on the blue curve (PD). It shows that if the smaller α is, the better error rate is. That is to say, if we consider more on the prior, the error rate is smaller. In addition, if α is greater or equal to one, the error rate goes to its maximum, not using the prior knowledge.

(b) **For D_1, compute the probability of error of the ML procedure identical to what we have used last week. Compare with the results of a). Can you explain? See "what to hand in" below.**

Now, we consider

$P_{X|T}(x|D) = G(x, \mu_{ML}, \sum)$, where μ_{ML} is the mean of the class (FG, BG).

Once again, after we have the posterior, we apply the BDR to predict the label given a pixel. Therefore, the predicted label of a pixel is the foreground if

$$log(P_{X|T}(x|D_{FG})\,P_T(D_{FG})) \geq log(P_{X|T}(x|D_{BG})\,P_T(D_{BG}))$$

Otherwise, the predicted label is background.

The curve of the ML is shown in Figure 1, Strategy 1 Dataset 1. Now, we only focus on the red line (ML). It shows that α will not influence the error rate since we only use the class mean as the parameter rather than different weight on the prior. We can also see that the performance on the PD is better than that on the ML.

(c) **Repeat a) with the MAP estimate of μ, i.e. using**

$$P_{x|T(x|D_1)} = P_{x|\mu}(x|\mu_{MAP}).$$

where

$$\mu_{MAP} = \underset{\mu}{argmax} P_{\mu|T}(\mu|D_1)$$

Compare the curve with those obtained above. Can you explain the results? See "what to hand in" below.

Now, we consider

$$P_{X|T}(x|D) = G(x, \mu_{MAP}, \sum)$$

$$\text{The mean } \mu_{MAP} = \mu_1$$

Once again, after we have the posterior, we apply the BDR to predict the label given a pixel. Therefore, the predicted label of a pixel is the foreground if

$$log(P_{X|T}(x|D_{FG})\, P_T(D_{FG})) \geq log(P_{X|T}(x|D_{BG})\, P_T(D_{BG}))$$

Otherwise, the predicted label is background.

The curve of the MAP is shown in Figure 1, Strategy 1 Dataset 1. Now, we only focus on the yellow curve (MAP). It shows that if the smaller α is, the better error rate is. That is to say, if we consider more on the prior, the error rate is smaller. The tendency looks the same as the PD. In addition, if α is greater or equal to one, the error rate of MAP goes to its maximum, and reaches the error rate of ML.

(d) **Repeat a) to c) for each of the datasets $D_i, i = 2, ..., 4$. Can you explain the results? See "what to hand in" below.**

By using the different D_i, which shown in the left half of Figure 1, we can see some difference. The observation are the followings:

- When α equals to or larger than one, MAP equals to ML. And if the α is samller, which means we use more priors, the performance is better.
- When we have more sample points in the dataset, the error rate decreases. However, although D_2 has smaller sample data than D_3 and D_4, the error rate in D_2 is better than those of D_3 and D_4. As a result, I think that there exists other factors influencing the performance of the model other than the size of the dataset, such as how good the features we get.

(e) **Repeat a) to d) under strategy 2 for the selection of the prior parameters. Comment the differences between the results obtained with the two strategies. See "what to hand in" below.**

By using the strategy 2, which shown in the right half of Figure 1, we can see a reverse results compared with using the strategy 1. The observation are the followings:

- When α equals to or larger than one, MAP equals to ML.
- When we have more sample points in the dataset, the error rate decreases.
- The results indicate that with larger α, we can get better values. That is to say, if we depend on less priors, we can get better values. The priors this time seems not as good as the one in the strategy 1. Also, the error rate in D_2 is better than those of D_3 and D_4. There may exist other factors influencing the performance of the model other than the size of the dataset as the reason mentioned above.

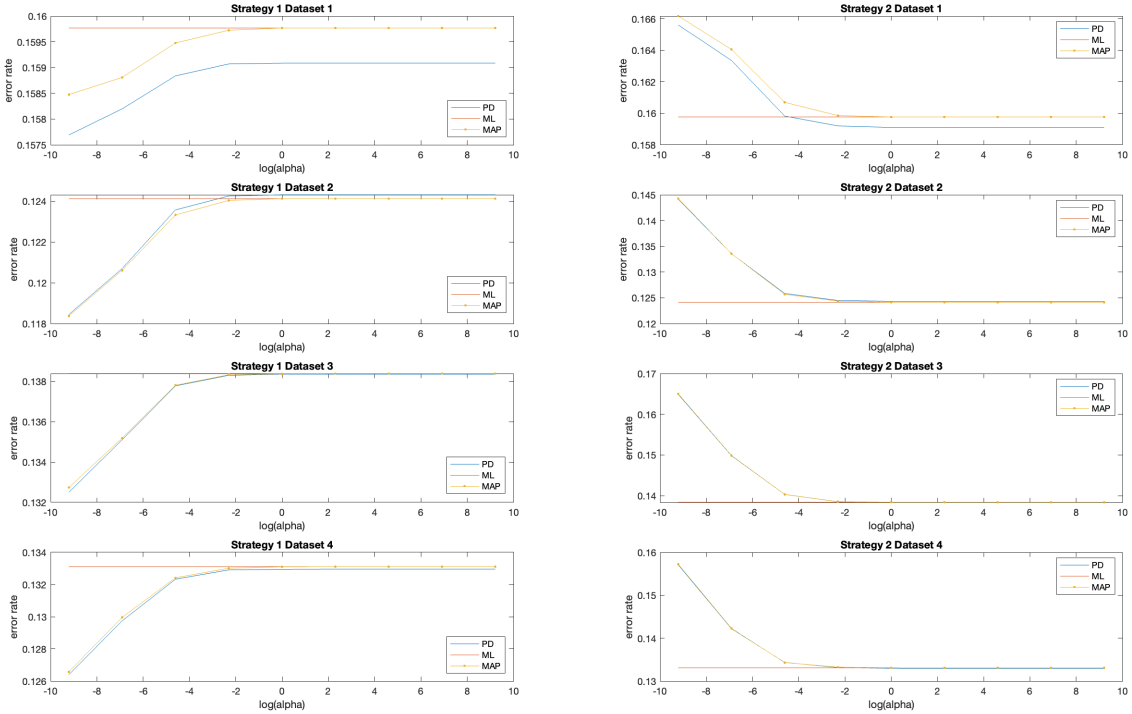

Figure 1: Error rate curve.

Code

```
%%%%%%%%%% Load data. %%%%%%%%%%

% Read img.
img = imread('hw3Data/cheetah.bmp');
img = double(img) / 255;
mask = imread('cheetah_mask.bmp');

% Padding img with zero.
I = zeros(263, 278);
for row = 5 : 259
    for col = 5 : 274
        I(row, col) = img(row - 4, col - 4);
    end
end

% Read pattern.
pattern = readmatrix('hw3Data/Zig-Zag Pattern.txt');

% Read training data.
load('hw3Data/TrainingSamplesDCT_subsets_8.mat');

load('hw3Data/Prior_1.mat');
w01 = W0;
mu0FG1 = transpose(mu0_FG);
mu0BG1 = transpose(mu0_BG);

load('hw3Data/Prior_2.mat')
w02 = W0;
mu0FG2 = transpose(mu0_FG);
mu0BG2 = transpose(mu0_BG);

clear W0;
clear mu0_FG;
```

Figure 2: Load data.

```
clear mu0_BG;

load('hw3Data/Alpha.mat')

% Choose S.
for s = 1 : 2
    if s == 1
        wS = w01;
        muSFG = mu0FG1;
        muSBG = mu0BG1;
    else
        wS = w02;
        muSFG = mu0FG2;
        muSBG = mu0BG2;
    end

    % Choose D.
    for d = 1 : 4
        if d == 1
            DFG = D1_FG;
            DBG = D1_BG;

        elseif d == 2
            DFG = D2_FG;
            DBG = D2_BG;

        elseif d == 3
            DFG = D3_FG;
            DBG = D3_BG;

        else
            DFG = D4_FG;
            DBG = D4_BG;
```

Figure 3: Choose different strategy and dataset.

```
67          end
68
69          % Compute covariance.
70          fSize = size(DFG);
71          bSize = size(DBG);
72          fCov = cov(DFG);
73          bCov = cov(DBG);
74          fMean = mean(DFG, 1);
75          bMean = mean(DBG, 1);
76          fCovInv = inv(fCov);
77          bCovInv = inv(bCov);
78          fCovDet = det(fCov);
79          bCovDet = det(bCov);
80
81          % Compute cov0.
82          cov0 = zeros(length(alpha), 64, 64);
83          for i = 1 : length(alpha)
84              for row = 1 : 64
85                  for col = 1 : 64
86                      if row == col
87                          cov0(i, row, col) = alpha(i) * wS(1, row);
88                      end
89                  end
90              end
91          end
92
93          % Compute mu1, cov1.
94          totalMuFG = zeros(length(alpha), 64, 1);
95          totalMuBG = zeros(length(alpha), 64, 1);
96          totalCovFG = zeros(length(alpha), 64, 64);
97          totalCovBG = zeros(length(alpha), 64, 64);
98
99          for i = 1 : length(alpha)
100             tmp = squeeze(cov0(i, :, :));
101             muFG = tmp * inv(tmp + fCov / fSize(1)) * transpose(fMean) + ...
102                 1 / fSize(1) * fCov * inv(tmp + fCov / fSize(1)) * muSFG;
103             muBG = tmp * inv(tmp + bCov / bSize(1)) * transpose(bMean) + ...
104                 1 / bSize(1) * bCov * inv(tmp + bCov / bSize(1)) * muSBG;
105
106             covFG = tmp * inv(tmp + fCov / fSize(1)) * fCov / fSize(1);
107             covBG = tmp * inv(tmp + bCov / bSize(1)) * bCov / bSize(1);
108
109             totalMuFG(i, :, :) = muFG;
110             totalMuBG(i, :, :) = muBG;
111             totalCovFG(i, :, :) = covFG;
112             totalCovBG(i, :, :) = covBG;
113         end
114
115         % Compute the predictive distribution.
116         totalCovPdFG = zeros(length(alpha), 64, 64);
117         totalCovPdBG = zeros(length(alpha), 64, 64);
118
119         for i = 1 : length(alpha)
120             for row = 1 : 64
121                 for col = 1 : 64
122                     totalCovPdFG(i, row, col) = totalCovFG(i, row, col) + .
123                         fCov(row, col);
124                     totalCovPdBG(i, row, col) = totalCovBG(i, row, col) + .
125                         bCov(row, col);
126                 end
127             end
128         end
129
130         % Compute prior.
131         priorFG = fSize / (fSize + bSize);
132         priorBG = bSize / (fSize + bSize);
```

Figure 4: Compute different covariance, mean, etc.

```
133
134         % Predict.
135         all1 = zeros(length(alpha), 255, 270);
136         all2 = zeros(length(alpha), 255, 270);
137         all3 = zeros(length(alpha), 255, 270);
138
139         allErrorRate1 = zeros(1, length(alpha));
140         allErrorRate2 = zeros(1, length(alpha));
141         allErrorRate3 = zeros(1, length(alpha));
142
143         for i = 1 : length(alpha)
144             covPdFGInv = inv(squeeze(totalCovPdFG(i, :, :)));
145             covPdBGInv = inv(squeeze(totalCovPdBG(i, :, :)));
146             covPdFGDet = det(squeeze(totalCovPdFG(i, :, :)));
147             covPdBGDet = det(squeeze(totalCovPdBG(i, :, :)));
148
149             for row = 1 : 255
150                 for col = 1 : 270
151                     block = zeros(8, 8);
152                     % Get the blcok.
153                     for r = row : row + 7
154                         for c = col : col + 7
155                             block(r - row + 1, c - col + 1) = I(r, c);
156                         end
157                     end
158
159                     % Compute DCT.
160                     dct2Block = dct2(block);
161                     flatBlock = zeros(1, 64);
162
163                     for r = 1 : 8
164                         for c = 1 : 8
165                             flatBlock(1, pattern(r, c) + 1) = dct2Block(r, c);
167                     end
168
169                     % PD: Use BDR to find class Y for each block.
170                     fRes = -0.5 * (flatBlock - totalMuFG(i, :)) * covPdFGInv * ...
171                         transpose(flatBlock - totalMuFG(i, :)) - ...
172                     0.5 * log((2 * pi) ^ 64 * covPdFGDet) + log(priorFG);
173
174                     bRes = -0.5 * (flatBlock - totalMuBG(i, :)) * covPdBGInv * ...
175                         transpose(flatBlock - totalMuBG(i, :)) - ...
176                     0.5 * log((2 * pi) ^ 64 * covPdBGDet) + log(priorBG);
177
178                     % Create a binary mask;
179                     if fRes >= bRes
180                         all1(i, row, col) = 1;
181                     else
182                         all1(i, row, col) = 0;
183                     end
184
185                     % ML: Use BDR to find class Y for each block.
186                     fRes = -0.5 * (flatBlock - fMean) * fCovInv * transpose(flatBlock - fMean) - ...
187                     0.5 * log((2 * pi) ^ 64 * fCovDet) + log(priorFG);
188
189                     bRes = -0.5 * (flatBlock - bMean) * bCovInv * transpose(flatBlock - bMean) - ...
190                     0.5 * log((2 * pi) ^ 64 * bCovDet) + log(priorBG);
191
192                     % Create a binary mask;
193                     if fRes >= bRes
194                         all2(i, row, col) = 1;
195                     else
196                         all2(i, row, col) = 0;
197                     end
198
199                     % Use BDR to find class Y for each block.
200                     fRes = -0.5 * (flatBlock - totalMuFG(i, :)) * fCovInv * ...
201                         transpose(flatBlock - totalMuFG(i, :)) - ...
202                         0.5 * log((2 * pi) ^ 64 * fCovDet) + log(priorFG);
203
204                     bRes = -0.5 * (flatBlock - totalMuBG(i, :)) * bCovInv * ...
205                         transpose(flatBlock - totalMuBG(i, :)) - ...
206                         0.5 * log((2 * pi) ^ 64 * bCovDet) + log(priorBG);
207
208                     % Create a binary mask;
209                     if fRes >= bRes
210                         all3(i, row, col) = 1;
211                     else
212                         all3(i, row, col) = 0;
213                     end
214                 end
215             end
216
217             errorCount1 = 0;
218             errorCount2 = 0;
219             errorCount3 = 0;
220
221             sizes = size(mask);
222             rows = sizes(1);
223             cols = sizes(2);
224
225             % Check whether the predicted label equals to the ground truth label.
226             for row  = 1 : rows
227                 for col  = 1 : cols
228                     if (mask(row, col) / 255 ~= all1(i, row, col))
229                         errorCount1 = errorCount1 + 1;
230                     end
231
232                     if (mask(row, col) / 255 ~= all2(i, row, col))
```

Figure 5: Compute predictive distribution, ML, MAP.

```
                    if (mask(row, col) / 255 ~= all3(i, row, col))
                        errorCount3 = errorCount3 + 1;
                    end
                end
            end

            allErrorRate1(1, i) = errorCount1 / (rows * cols);
            allErrorRate2(1, i) = errorCount2 / (rows * cols);
            allErrorRate3(1, i) = errorCount3 / (rows * cols);

            disp('%%%%%%%%%%');
            disp(allErrorRate1(1, i));
            disp(allErrorRate2(1, i));
            disp(allErrorRate3(1, i));
            disp(s);
            disp(d);
            disp(i);
            disp('%%%%%%%%%%');
        end

        % Plot the error rate curve.
        x = zeros(1, length(alpha));

        for i = 1 : length(alpha)
            x(1, i) = log(alpha(1, i));
        end

        % Compared with all curves.
        subplot(4, 2, (d - 1) * 2 + s)
        plot(x, allErrorRate1, x, allErrorRate2, x, allErrorRate3, '.-'), legend('PD', 'ML', 'MAP');
        title(['Strategy ', num2str(s), ' Dataset ', num2str(d)]);
        xlabel('log(alpha)');
        ylabel('error rate') ;
    end
end

savefig('allErrorRate_3_4.fig');
clf;
```

Figure 6: Compute error rate and plot the gram.