

# EE583 Pattern Recognition HW2

Kutay Uğurlu 2232841

November 21, 2021

## 1 Question 1

Maximum Likelihood Estimate of Multivariate Gaussian random variable are calculated as:

- $\hat{\mu} = \frac{1}{m} \sum_{i=1}^m X^{(i)}$
- $\hat{\Sigma} = \frac{1}{m} \sum_{i=1}^m (X^{(i)} - \hat{\mu})(X^{(i)} - \hat{\mu})^T$

where  $X^{(i)}$  is the  $i^{th}$  observation.

The given MATLAB code in Section 6.1 produces the following results:

$$\begin{array}{lll} m = 10 & \hat{\mu} = [-0.4519 & 1.0984] & \hat{\Sigma} = \begin{bmatrix} 0.3191 & 0.1799 \\ 0.1799 & 0.6506 \end{bmatrix} \\ m = 1000 & \hat{\mu} = [-0.7367 & 0.5176] & \hat{\Sigma} = \begin{bmatrix} 0.4730 & 0.2925 \\ 0.2925 & 0.8079 \end{bmatrix} \end{array}$$

It is observed that the estimations get more accurate, *i.e.* the calculated metrics get closer to the original ones, with increasing number of samples.

## 2 Question 2

The ML estimate is calculated using mean of the generated data. On the other hand, to calculate the MAP estimate of  $\mu$ , following formula is utilized:

$$\hat{\mu}_n = \frac{n\sigma_0^2}{n\sigma_0^2 + \sigma^2} \frac{1}{n} \sum_{i=1}^n x_i + \frac{\sigma^2 \mu_0}{n\sigma_0^2 + \sigma^2} \quad (1)$$

The formula given is a normalized weighted average of mean of the training data and mean assumed previously. The mean estimate in Eq. 1 converges to sample mean when  $n \rightarrow \infty$ . For  $n = 0$ , the estimate is just the prior mean, and it gets closer to the sample mean when  $n$  increases. Therefore, one would expect to see an estimate around  $\mu_0$  for "small"  $n$ . However, even for small values of  $n$ , such as 25, the weight of the sample mean approaches to 0.95. Still, for small  $n$ , the sample mean does not converge to the actual sample mean, and in one experiment, I have made the following observations.

### 2.1 25 Samples

$$\hat{x}_{ML} = 2.8995 \quad \hat{x}_{MAP} = 2.9019$$

### 2.2 1000 samples

$$\hat{x}_{ML} = 2.9983 \quad \hat{x}_{MAP} = 2.9984$$

With increasing  $n$ , the sample mean dominates the estimate for sure. But for small  $n$  around 20, the estimate is highly dependent on the experimental ensemble average. In some experiments, this value does not converge to the expected value of the random variable. It is also observed that, the estimates gets closer with increasing number of training samples.

### 3 Question 3

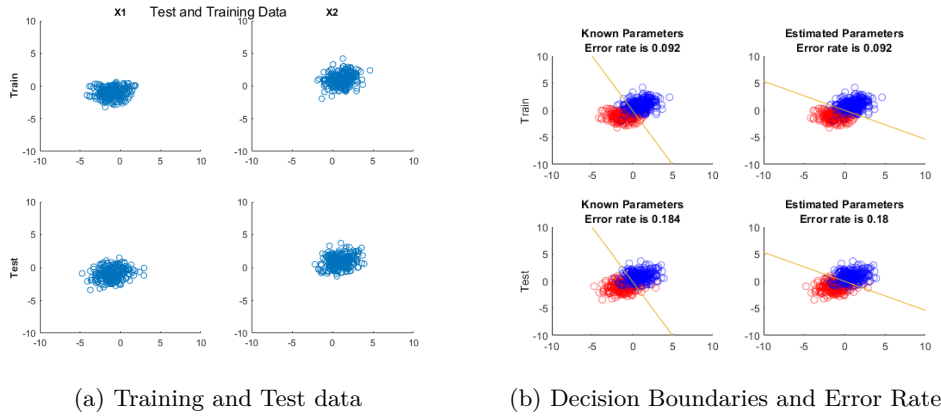


Figure 1: Data distributions and Decision Boundaries

As expected, training accuracy is higher than the test accuracy. Furthermore, decision boundaries obtained with estimated parameters resulted in lower accuracy, since the minimum error rate classifier boundaries are calculated analytically with already known distribution parameters.

### 4 Question 4

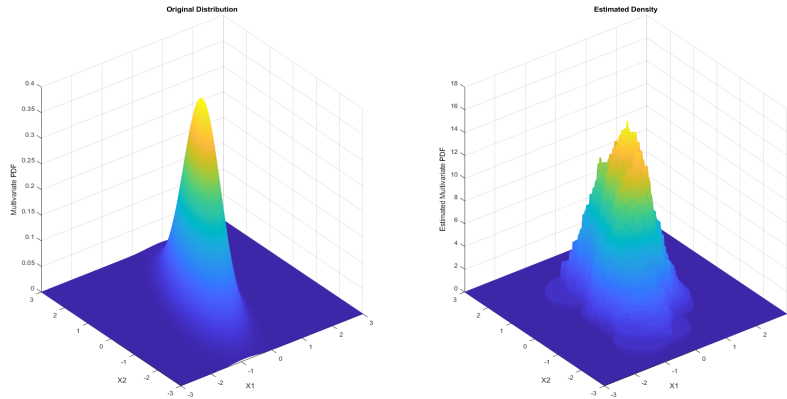
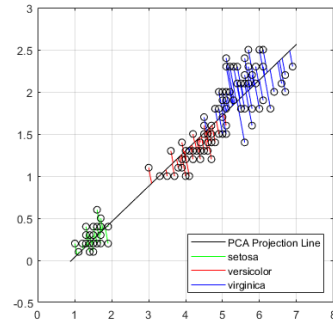


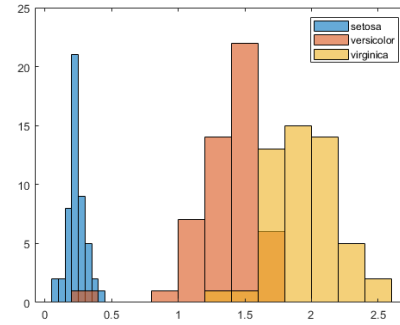
Figure 2: Parzen window estimation results

The selection of initial volume and volume shrinking formula are presented in subsection 6.4.

## 5 Question 5




(a) PCA Projection onto 1D



(b) Histograms

Figure 3: PCA Projection and histogram

## 6 APPENDIX

The code given in this section is shared @.

### 6.1 Q1

```
1 % Q1
2 m = 1000;
3 sigma_1 = [0.5 0.3;0.3 0.8];
4 mu_1 = [-0.75;0.5];
5 X = mvnrnd(mu_1,sigma_1,m);
6 mu_1_hat = mean(X);
7
8 sum = zeros(size(sigma_1));
9 for i = 1:m
10     sum = sum + (X(i,:) - mu_1_hat)' * (X(i,:) - mu_1_hat);
11 end
12 sigma_1_hat = sum/m;
13 mu_1_hat
14 sigma_1_hat
```

**6.2 Q2**

```
1 % Part i – ML Estimate of mean
2 n = 5;
3 sigma_sq = 0.49;
4 X = sqrt(sigma_sq) * randn(n,1) + 3;
5 mu_x_ML = mean(X);
6 % Part ii – MAP Estimate of mean
7 mu_0 = 2.8;
8 sigma_0_sq = 0.8;
9 mu_n = n*sigma_0_sq/(n*sigma_0_sq+sigma_sq) * mean(X) + sigma_sq
    /(n*sigma_0_sq+sigma_sq) * mu_0;
```

## 6.3 Q3

```

1 m = 250;
2 sigma = [1.4 0.2; 0.2 0.8];
3 mu_1 = [-1; -1];
4 X1 = mvnrnd(mu_1, sigma, m);
5 mu_2 = [1; 1];
6 X2 = mvnrnd(mu_2, sigma, m);
7 edgealpha = 0.5;
8 %% Decision boundary
9 % Since class priors are equal
10 figure
11 subplot(2,2,1)
12 x0 = 0.5 * (mu_1 + mu_2); % = 0
13 A = (mu_1 - mu_2)' * (inv(sigma)); % = -1.1111 -2.2222
14 % Then decision boundary  $\rightarrow -x_1 - 2x_2 = 0 \rightarrow x_1 = -2x_2$ 
15 scatter(X1(:,1), X1(:,2), 'red', 'MarkerEdgeAlpha', edgealpha)
16 hold on
17 scatter(X2(:,1), X2(:,2), 'blue', 'MarkerEdgeAlpha', edgealpha)
18 hold on
19 x = linspace(-5, 5, 100);
20 plot(x, -2*x)
21 ylabel('Train')
22 xlim([-10, 10])
23 ylim([-10, 10])
24 conf_mat_1 = zeros(2,2);
25
26 for i = 1:m
27     x = X1(i,:);
28     eval_at_line = A * transpose(x);
29     if eval_at_line > 0
30         conf_mat_1(1,1) = conf_mat_1(1,1) + 1;
31     else
32         conf_mat_1(2,1) = conf_mat_1(2,1) + 1;
33     end
34
35     x = X2(i,:);
36     eval_at_line = A * transpose(x);
37     if eval_at_line > 0
38         conf_mat_1(1,2) = conf_mat_1(1,2) + 1;
39     else
40         conf_mat_1(2,2) = conf_mat_1(2,2) + 1;
41     end
42 end
43 error_rate_1 = (conf_mat_1(1,2) + conf_mat_1(2,1)) / (2*m);

```

```

44
45 title({'Known Parameters',[ 'Error rate is ',num2str(error_rate_1)
    ]})
46
47 %% ML Estimation
48 subplot(2,2,2)
49 [mu_1_hat,Sigma_1_hat] = estimate_ML(X1);
50 [mu_2_hat,Sigma_2_hat] = estimate_ML(X2);
51 Sigma = 0.5 * (Sigma_1_hat+Sigma_2_hat);
52 x0 = 0.5 * (mu_1_hat+mu_2_hat); % = 0
53 A_ML = (mu_1_hat-mu_2_hat)'*(inv(Sigma)); %
54 syms f(x);
55 scatter(X1(:,1),X1(:,2),'red','MarkerEdgeAlpha',edgealpha)
56 hold on
57 scatter(X2(:,1),X2(:,2),'blue','MarkerEdgeAlpha',edgealpha)
58 hold on
59 f(x) = x0(2) - A_ML(1)/A_ML(2) * (x-x0(1));
60 fplot(f(x))
61 xlim([-10,10])
62 ylim([-10,10])
63
64 conf_mat_2 = zeros(2,2);
65
66 for i = 1:m
67     x = X1(i,:);
68     eval_at_line = A * transpose(x);
69     if eval_at_line > 0
70         conf_mat_2(1,1) = conf_mat_2(1,1) + 1;
71     else
72         conf_mat_2(2,1) = conf_mat_2(2,1) + 1;
73     end
74
75     x = X2(i,:);
76     eval_at_line = A * transpose(x);
77     if eval_at_line > 0
78         conf_mat_2(1,2) = conf_mat_2(1,2) + 1;
79     else
80         conf_mat_2(2,2) = conf_mat_2(2,2) + 1;
81     end
82 end
83 error_rate_2 = (conf_mat_2(1,2) + conf_mat_2(2,1)) / (2*m);
84 title({'Estimated Parameters',[ 'Error rate is ',num2str(
    error_rate_2)]})
85
86

```



```

87 %% Test Data
88
89 X1_test = mvnrnd(mu_1,sigma,m);
90 X2_test = mvnrnd(mu_2,sigma,m);
91
92 %% Test Data Known
93 subplot(2,2,3)
94
95 x0 = 0.5 * (mu_1+mu_2); % = 0
96 A = (mu_1-mu_2)'*(inv(sigma)); % = -1.1111 -2.2222
97 % Then decision boundary -> -x1-2x2 = 0 -> x1 = -2x2
98 scatter(X1_test(:,1),X1_test(:,2),'red','MarkerEdgeAlpha',
          edgealpha)
99 hold on
100 scatter(X2_test(:,1),X2_test(:,2),'blue','MarkerEdgeAlpha',
          edgealpha)
101 hold on
102 x = linspace(-5,5,100);
103 plot(x,-2*x)
104 ylabel('Test')
105 xlim([-10,10])
106 ylim([-10,10])
107 conf_mat_1 = zeros(2,2);
108
109 for i = 1:m
110     x = X1_test(i,:);
111     eval_at_line = A * transpose(x);
112     if eval_at_line > 0
113         conf_mat_1(1,1) = conf_mat_1(1,1) + 1;
114     else
115         conf_mat_1(2,1) = conf_mat_1(2,1) + 1;
116     end
117
118     x = X2_test(i,:);
119     eval_at_line = A * transpose(x);
120     if eval_at_line > 0
121         conf_mat_1(1,2) = conf_mat_1(1,2) + 1;
122     else
123         conf_mat_1(1,2) = conf_mat_1(2,2) + 1;
124     end
125 end
126 error_rate_1 = (conf_mat_1(1,2) + conf_mat_1(2,1)) / (2*m);
127
128 title({'Known Parameters',[ 'Error rate is ',num2str(error_rate_1)
          ]})

```

```

129
130 %% ML Estimation Test data
131 subplot(2,2,4)
132 scatter(X1_test(:,1),X1_test(:,2),'red','MarkerEdgeAlpha',
         edgealpha)
133 hold on
134 scatter(X2_test(:,1),X2_test(:,2),'blue','MarkerEdgeAlpha',
         edgealpha)
135 hold on
136 syms f(x)
137 f(x) = x0(2) - A_ML(1)/A_ML(2) * (x-x0(1));
138 fplot(f(x))
139 xlim([-10,10])
140 ylim([-10,10])
141
142 conf_mat_2 = zeros(2,2);
143
144 for i = 1:m
145     x = X1_test(i,:);
146     eval_at_line = A_ML * transpose(x);
147     if eval_at_line > 0
148         conf_mat_2(1,1) = conf_mat_2(1,1) + 1;
149     else
150         conf_mat_2(2,1) = conf_mat_2(2,1) + 1;
151     end
152
153     x = X2_test(i,:);
154     eval_at_line = A_ML * transpose(x);
155     if eval_at_line > 0
156         conf_mat_2(2,2) = conf_mat_2(1,2) + 1;
157     else
158         conf_mat_2(1,2) = conf_mat_2(2,2) + 1;
159     end
160 end
161 error_rate_2 = (conf_mat_2(1,2) + conf_mat_2(2,1)) / (2*m);
162 title({'Estimated Parameters',[ 'Error rate is ',num2str(
         error_rate_2)]})
163
164 figure
165 subplot(2,2,1)
166 subplot(2,2,1)
167 scatter(X1(:,1),X1(:,2))
168 title('X1')
169 xlim([-10,10])
170 ylim([-10,10])

```

```
171 ylabel('Train', 'FontWeight', 'bold')
172 subplot(2,2,2)
173 scatter(X2(:,1), X2(:,2))
174 title('X2')
175 xlim([-10,10])
176 ylim([-10,10])
177 subplot(2,2,3)
178 scatter(X1_test(:,1), X1_test(:,2))
179 xlim([-10,10])
180 ylim([-10,10])
181 ylabel('Test', 'FontWeight', 'bold')
182 subplot(2,2,4)
183 scatter(X2_test(:,1), X2_test(:,2))
184 xlim([-10,10])
185 ylim([-10,10])
```

## 6.4 Q4

```

1 m = 50;
2 mu = [0 0];
3 Sigma = [0.25 0.3; 0.3 1];
4 x1 = -3:0.02:3;
5 x2 = -3:0.02:3;
6 [X1,X2] = meshgrid(x1,x2);
7 X = [X1(:) X2(:)];
8 y = mvnpdf(X,mu,Sigma);
9 y = reshape(y,length(x2),length(x1));
10 subplot(1,2,1)
11 surf(x1,x2,y,'EdgeColor','interp','EdgeAlpha',1)
12 title('Original Distribution')
13 xlabel('X1')
14 ylabel('X2')
15 zlabel('Multivariate PDF')
16
17 %% Parzen
18 mu = [0 0];
19 Sigma = [0.25 0.3; 0.3 1];
20 n = 50;
21 X_sampled = mvnrnd(mu, Sigma, n);
22 h = 1.55;
23 V = h/n;
24 pn = zeros(size(X,1),1);
25 Estimated_density = zeros(size(X,1),1);
26 for i = 1:size(X,1)
27     x = X(i,:);
28     sum = 0;
29     for j = 1:n
30         sum = sum + 1/V * parzen_win((x-X_sampled(j,:))/h);
31     end
32     pn = 1/n * sum;
33     Estimated_density(i) = pn;
34 end
35 subplot(1,2,2)
36 Estimated_density = reshape(Estimated_density,length(x2),length(
    x1));
37 surf(x1,x2,Estimated_density,'EdgeColor','interp','EdgeAlpha',1)
38 title('Estimated Density')
39 xlabel('X1')
40 ylabel('X2')
41 zlabel('Estimated Multivariate PDF')

```

## 6.5 Q5

```

1 X = load('fisheriris');
2 features = X.meas;
3 classes = X.species;
4 X = features(:, [3 4]);
5 figure
6 scatter(X(:,1), X(:,2), 'bo');
7 grid on;
8 maxlim = max(abs(X(:)))*1.1;
9 axis([-maxlim maxlim -maxlim maxlim]);
10
11 %% PCA
12 [coeff, score, roots] = pca(X);
13 basis = coeff(:,1);
14 normal = coeff(:,2);
15 [n,p] = size(X);
16 meanX = mean(X,1);
17 Xfit = repmat(meanX,n,1) + score(:,1)*coeff(:,1)';
18 residuals = X - Xfit;
19
20 %% Visualize Line Fit
21 dirVect = coeff(:,1);
22 Xfit1 = repmat(meanX,n,1) + score(:,1)*coeff(:,1)';
23 t = [min(score(:,1))-.2, max(score(:,1))+.2];
24 endpts = [meanX + t(1)*dirVect'; meanX + t(2)*dirVect'];
25 plot(endpts(:,1), endpts(:,2), 'k-');
26 X1 = [X(:,1) Xfit1(:,1)];
27 X2 = [X(:,2) Xfit1(:,2)];
28 hold on
29 plot(X1(1:50,:), X2(1:50,:), 'g-', X(1:50,1), X(1:50,2), 'ko');
30 hold on
31 plot(X1(50:100,:), X2(50:100,:), 'r-', X(50:100,1), X(50:100,2), '
    ko');
32 hold on
33 plot(X1(100:150,:), X2(100:150,:), 'b-', X(100:150,1), X
    (100:150,2), 'ko');
34
35 L(1) = plot(nan, nan, 'k-');
36 L(2) = plot(nan, nan, 'g-');
37 L(3) = plot(nan, nan, 'r-');
38 L(4) = plot(nan, nan, 'b-');
39
40 legend_cell = cell(4,1);
41 legend_cell{1} = 'PCA Projection Line';

```

```
42 legend_cell{2} = 'setosa';
43 legend_cell{3} = 'versicolor';
44 legend_cell{4} = 'virginica';
45
46 legend(L, legend_cell, 'Location', 'southeast');
47
48 hold off
49 maxlim = max(abs(X(:)))*1.1;
50 axis([0 8 -0.5 3]);
51 axis square
52 grid on
53 setosa = Xfit1(1:50,2);
54 versicolor = Xfit1(50:100,2);
55 virginica = Xfit1(100:150,2);
56 figure
57 histogram(setosa)
58 hold on
59 histogram(versicolor)
60 hold on
61 histogram(virginica)
62 legend('setosa', 'versicolor', 'virginica')
```