

# HW 4

## Part 1:

Given  $Y$ , the MLE of  $\hat{x}$  is:

$$\begin{aligned}\hat{x} &= \arg \min_x (y - ax)^T \Sigma^{-1} (y - ax) + N \log(2\pi |\Sigma|) \\ &= \arg \min_x (y - ax)^T \Sigma^{-1} (y - ax) \\ &= \arg \min_x (Sy - Sax)^T (Sy - Sax) \\ &= \arg \min_x \|Sy - Sax\|^2\end{aligned}$$

Note that:  $S = \sqrt{\Sigma^{-1}}$

$$\begin{aligned}\hat{x} &= \arg \max_x \|Sy - Sax\|^2 \quad ; \quad V = Sy, \quad M = Sa \\ \hat{x} &= \arg \max_x \|V - Mx\|^2\end{aligned}$$

take first order derivative:  $\frac{\partial f}{\partial x} = 0$ , when  $M^T(V - Mx) = 0$

$$\begin{aligned}\hat{x} &= (M^T M)^{-1} M^T V \\ &= \cancel{(a^T \Sigma^{-1} a)^{-1} a^T \Sigma^{-1} y} \\ &= (a^T \Sigma^{-1} a)^{-1} a^T \Sigma^{-1} y\end{aligned}$$

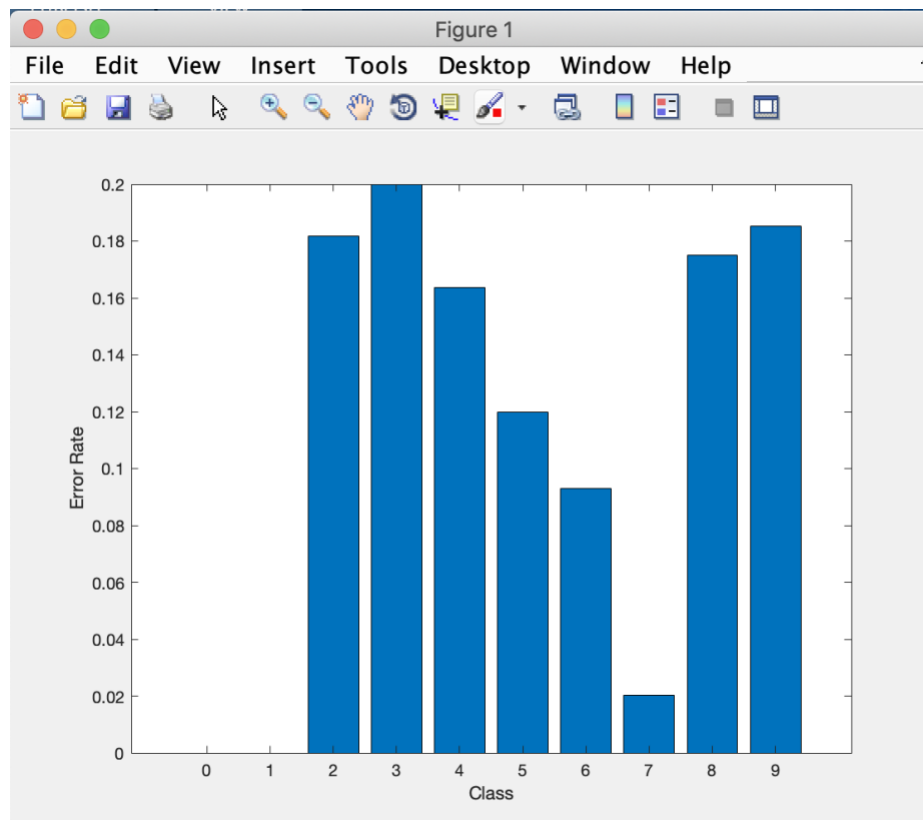
2<sup>nd</sup> order derivative:  $\frac{\partial^2 f}{\partial x^2} > 0$  holds, we have a minimum which is maximum of the log-likelihood.

Same logic applies ( $Y = a \cdot x$ ), substitute  $a$  with  $x$ :  $\hat{a} = (x^T x)^{-1} \cdot (x^T \cdot y)$

The maximum likelihood estimation of  $a$  can be represented by:  $(X^T * Y) * (X^T * X)^{-1}$

## Part 2:

After normalized all test images by dividing **a**, we implement classification using least square distance metric, and we got the flowing plot, table, and total error rate:

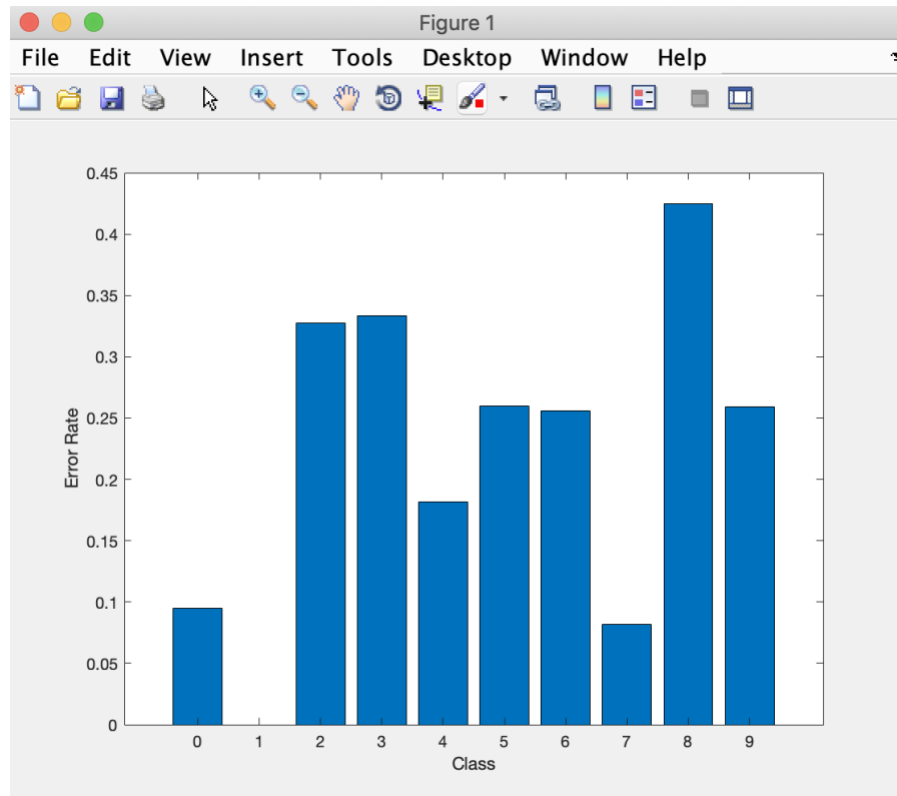


T					
10x5 table					
	1 Class	2 Total_num_of_images	3 Correctly_classified	4 Incorrectly_classified	5 Error_rate
1	'0'	42	42	0	0
2	'1'	67	67	0	0
3	'2'	55	45	10	0.1818
4	'3'	45	36	9	0.2000
5	'4'	55	46	9	0.1636
6	'5'	50	44	6	0.1200
7	'6'	43	39	4	0.0930
8	'7'	49	48	1	0.0204
9	'8'	40	33	7	0.1750
10	'9'	54	44	10	0.1852

total\_error\_rate 0.1120

### Part 3:

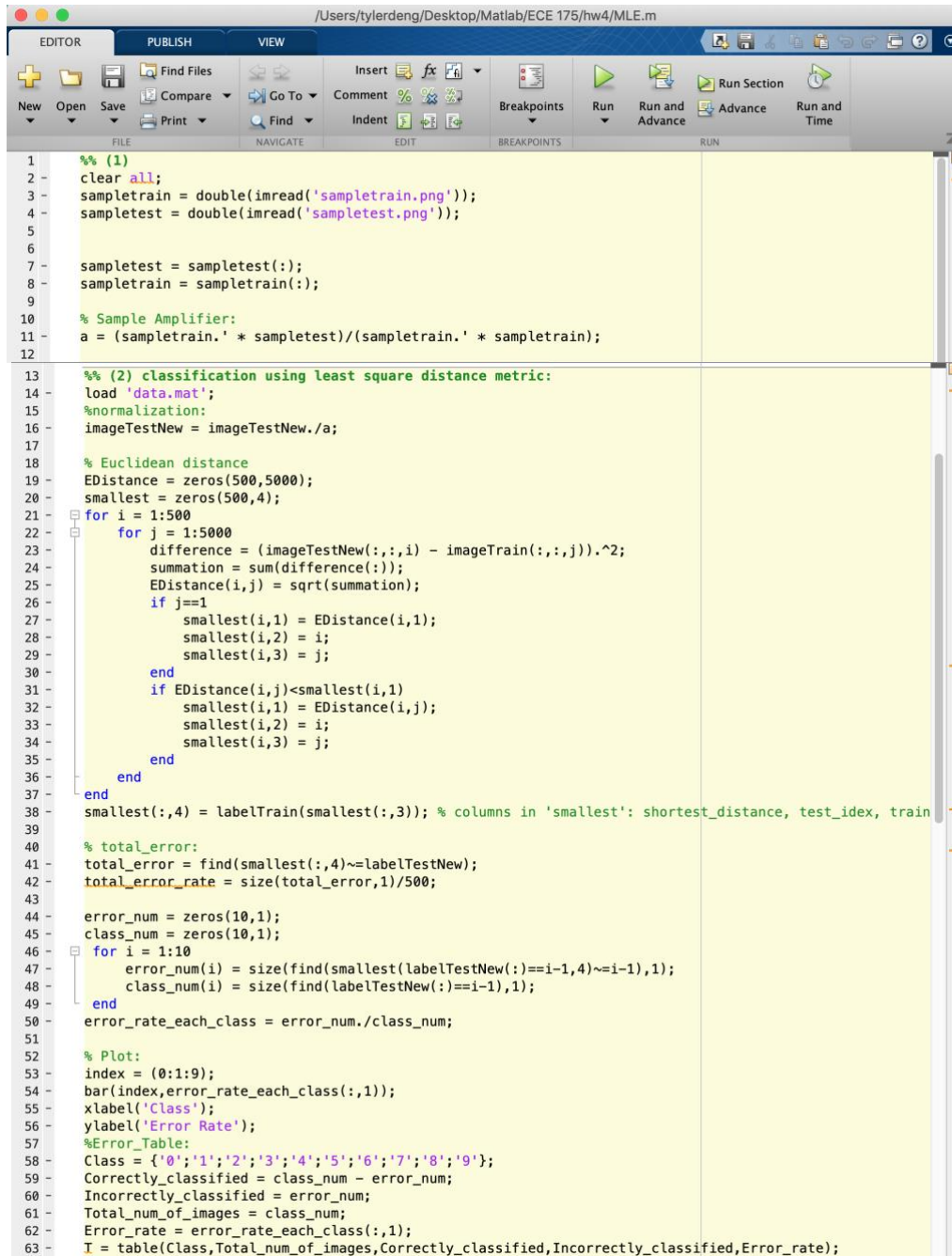
Doing NN classifier without normalization process, we got:



10x5 table					
	1 Class	2 Total_num_of_images	3 Correctly_classified	4 Incorrectly_classified	5 Error_rate
1	'0'	42	38	4	0.0952
2	'1'	67	67	0	0
3	'2'	55	37	18	0.3273
4	'3'	45	30	15	0.3333
5	'4'	55	45	10	0.1818
6	'5'	50	37	13	0.2600
7	'6'	43	32	11	0.2558
8	'7'	49	45	4	0.0816
9	'8'	40	23	17	0.4250
10	'9'	54	40	14	0.2593

total\_error\_rate 0.2120

# Appendix



```
1  %% (1)
2  clear all;
3  sampletrain = double(imread('sampletrain.png'));
4  sampletest = double(imread('sampletest.png'));
5
6
7  sampletest = sampletest(:);
8  sampletrain = sampletrain(:);
9
10 % Sample Amplifier:
11 a = (sampletrain.' * sampletest)/(sampletrain.' * sampletrain);
12
13 %% (2) classification using least square distance metric:
14 load 'data.mat';
15 %normalization:
16 imageTestNew = imageTestNew./a;
17
18 % Euclidean distance
19 EDistance = zeros(500,5000);
20 smallest = zeros(500,4);
21 for i = 1:500
22     for j = 1:5000
23         difference = (imageTestNew(:,i) - imageTrain(:,j)).^2;
24         summation = sum(difference(:));
25         EDistance(i,j) = sqrt(summation);
26         if j==1
27             smallest(i,1) = EDistance(i,1);
28             smallest(i,2) = i;
29             smallest(i,3) = j;
30         end
31         if EDistance(i,j)<smallest(i,1)
32             smallest(i,1) = EDistance(i,j);
33             smallest(i,2) = i;
34             smallest(i,3) = j;
35         end
36     end
37 end
38 smallest(:,4) = labelTrain(smallest(:,3)); % columns in 'smallest': shortest_distance, test_idx, train
39
40 % total_error:
41 total_error = find(smallest(:,4)~=labelTestNew);
42 total_error_rate = size(total_error,1)/500;
43
44 error_num = zeros(10,1);
45 class_num = zeros(10,1);
46 for i = 1:10
47     error_num(i) = size(find(smallest(labelTestNew(:)==i-1,4)~=i-1),1);
48     class_num(i) = size(find(labelTestNew(:)==i-1),1);
49 end
50 error_rate_each_class = error_num./class_num;
51
52 % Plot:
53 index = (0:1:9);
54 bar(index,error_rate_each_class(:,1));
55 xlabel('Class');
56 ylabel('Error Rate');
57 %Error_Table:
58 Class = {'0';'1';'2';'3';'4';'5';'6';'7';'8';'9'};
59 Correctly_classified = class_num - error_num;
60 Incorrectly_classified = error_num;
61 Total_num_of_images = class_num;
62 Error_rate = error_rate_each_class(:,1);
63 I = table(Class,Total_num_of_images,Correctly_classified,Incorrectly_classified,Error_rate);
```

```

68 %% (3) NN classifier without MLE
69 clear all;
70 load 'data.mat';
71 %normalization:
72 %imageTestNew = imageTestNew./a;
73 % Euclidean distance
74 EDistance = zeros(500,5000);
75 smallest = zeros(500,4);
76 for i = 1:500
77     for j = 1:5000
78         difference = (imageTestNew(:,i) - imageTrain(:,j)).^2;
79         summation = sum(difference(:));
80         EDistance(i,j) = sqrt(summation);
81         if j==1
82             smallest(i,1) = EDistance(i,1);
83             smallest(i,2) = i;
84             smallest(i,3) = j;
85         end
86         if EDistance(i,j)<smallest(i,1)
87             smallest(i,1) = EDistance(i,j);
88             smallest(i,2) = i;
89             smallest(i,3) = j;
90         end
91     end
92 end
93 smallest(:,4) = labelTrain(smallest(:,3)); % columns in 'smallest': shortest_distance, test_idx, train
94
95 % total_error:
96 total_error = find(smallest(:,4)~=labelTestNew);
97 total_error_rate = size(total_error,1)/500;
98
99 error_num = zeros(10,1);
100 class_num = zeros(10,1);
101 for i = 1:10
102     error_num(i) = size(find(smallest(labelTestNew(:)==i-1,4)~=i-1),1);
103     class_num(i) = size(find(labelTestNew(:)==i-1),1);
104 end
105 error_rate_each_class = error_num./class_num;
106
107 % Plot:
108 index = (0:1:9);
109 bar(index,error_rate_each_class(:,1));
110 xlabel('Class');
111 ylabel('Error Rate');
112 %Error_Table:
113 Class = {'0';'1';'2';'3';'4';'5';'6';'7';'8';'9'};
114 Correctly_classified = class_num - error_num;
115 Incorrectly_classified = error_num;
116 Total_num_of_images = class_num;
117 Error_rate = error_rate_each_class(:,1);
118 T = table(Class,Total_num_of_images,Correctly_classified,Incorrectly_classified,Error_rate);
119

```