

### ECE 271A Homework 3

1. This week we will continue trying to classify our cheetah example. Once again we use the decomposition into  $8 \times 8$  image blocks, compute the DCT of each block, and zig-zag scan. We also continue to assume that the class-conditional densities are multivariate Gaussians of 64 dimensions. The goal is to understand the benefits of a Bayesian solution.
  - (a) Consider the training set  $\mathbf{D}_1$  and **strategy 1**. For each class, compute the covariance  $\Sigma$  of the classconditional, and the posterior  $\mu_1$ , and  $\Sigma_1$  of  $P_{\mu|T}(\mu|D_1) = G(\mu, \mu_1, \Sigma_1)$ . 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  $\alpha$ . Repeat the procedure for the values of given in the file **Alpha.mat**. Plot the curve of the probability of error as a function of  $\alpha$ . Can you explain the results?

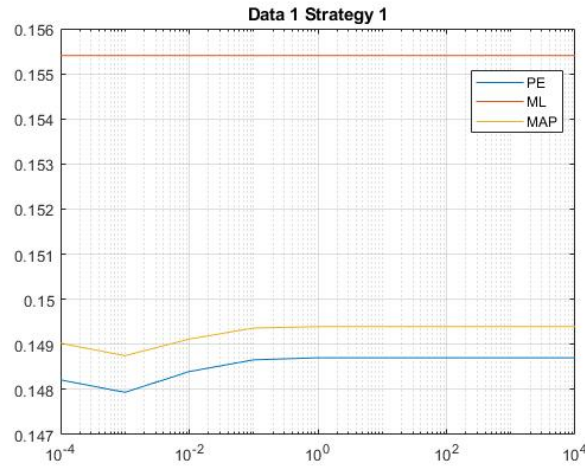
*Solution.*

For **Predictive Distribution method**, we get our training data sets from the prior predictive distribution, i.e., a collection of data sets generated from the model. After we have obtained the posterior distributions of the parameters, we can now use the posterior distributions to generate future data from the model. From the formula, we can see that the posterior distribution is a nothing more than a Gaussian distribution of  $\mu$ s, with  $\mu_n$  and  $\sigma_n^2$  unknown. However, we can obtain  $\mu_n$  and  $\sigma_n^2$  from the formulas mentioned in the lectures. It is worth noting that there is a crucial variable in the formula, i.e.,  $\sigma_0^2$ .  $\sigma_0^2$  is dependent upon our choice of  $\alpha$ . If we change the value of  $\alpha$ , then the value of  $\sigma_0^2$  will change accordingly. However, when the value of  $\alpha$  is greater than 1, the results only change subtly.  $\alpha$  acts as a weight, deciding how much we can trust our a priori data.

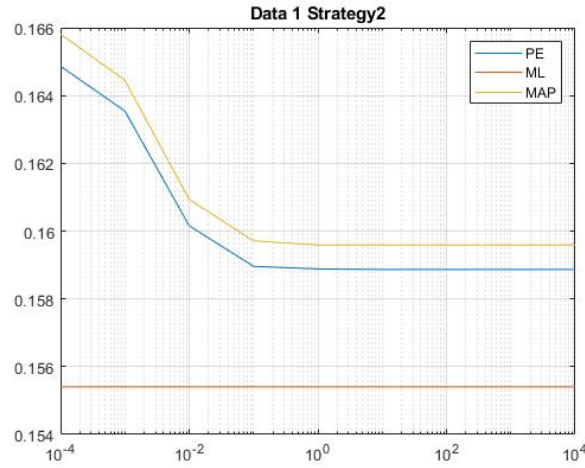
- (b) For  $D_1$ , compute the probability of error of the ML procedure identical to what we have used last week. Compare with the results of **a**. Can you explain? See "what to hand in" below.

*Solution.*

For **ML method**, the value of  $\alpha$  does not matter. Therefore, the error rates of **ML** are consistent. It should be stressed out that in some cases, the error rates of **ML** perform better than other methods. Such as *Data 1 Strategy 2*. In fact, throughout all 4 data with *Strategy 2*, i.e., *Data 1 Strategy 2*, *Data 2 Strategy 2*, etc, **ML** perform better.

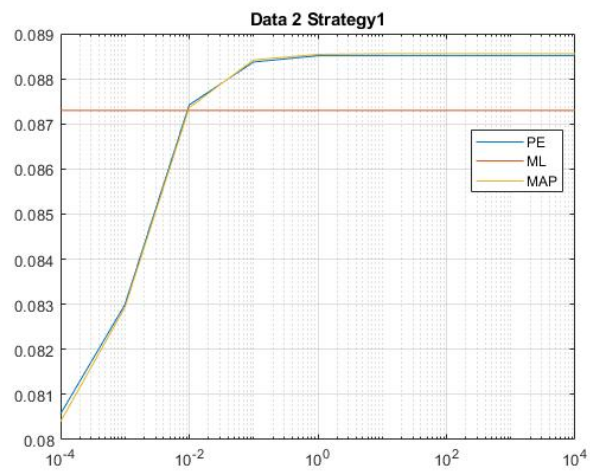


(a)

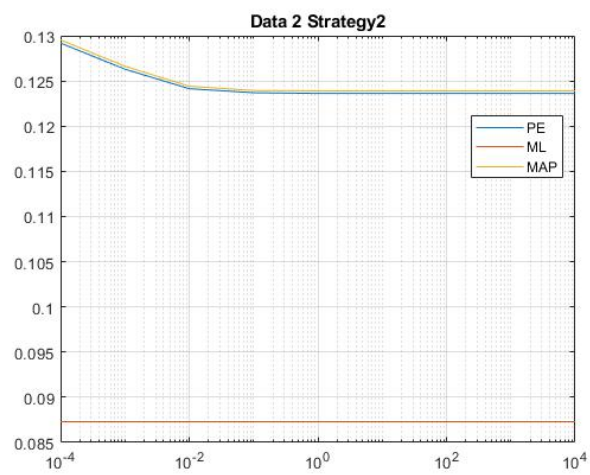


(b)

Figure 1: Error Rate From Data 1 (PE vs. MAP vs. ML): (a) Strategy 1 (b) Strategy 2

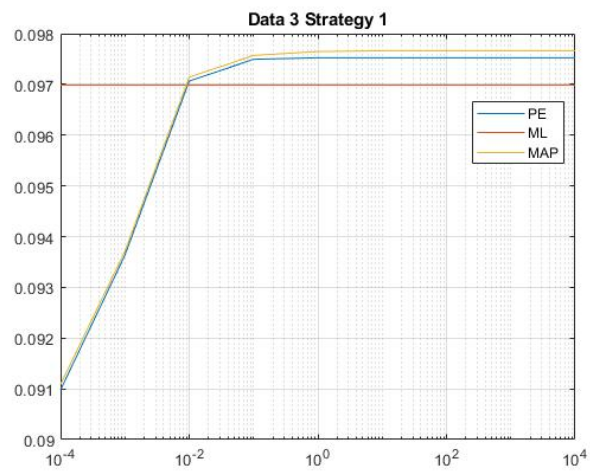


(a)

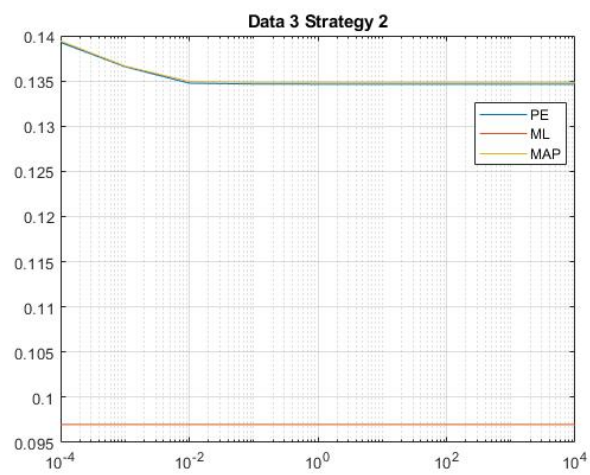


(b)

Figure 2: Error Rate From Data 2 (PE vs. MAP vs. ML): (a) Strategy 1 (b) Strategy 2

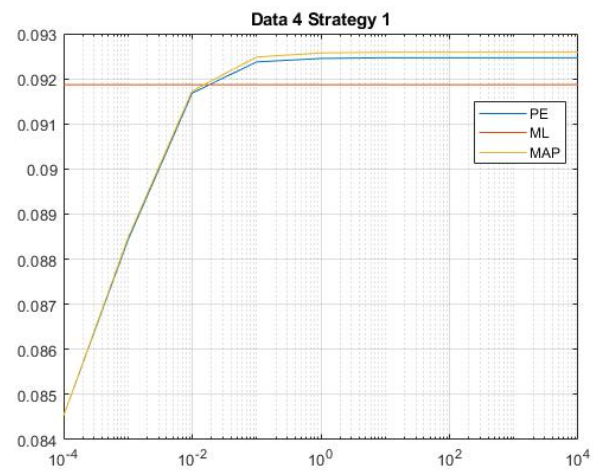


(a)

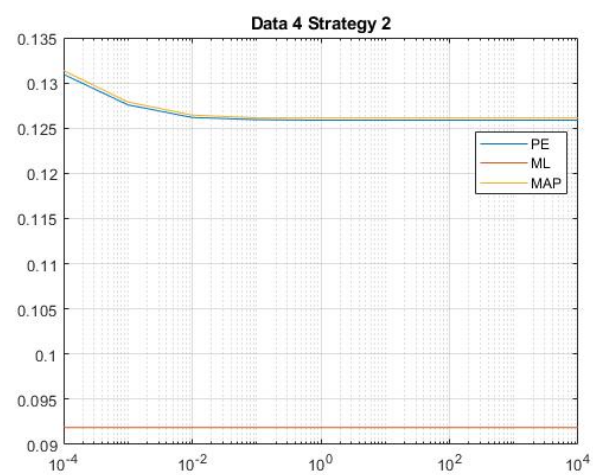


(b)

Figure 3: Error Rate From Data 3 (PE vs. MAP vs. ML): (a) Strategy 1 (b) Strategy 2



(a)



(b)

Figure 4: Error Rate From Data 4 (PE vs. MAP vs. ML): (a) Strategy 1 (b) Strategy 2

- (c) Repeat **a** with the MAP estimate of  $\mu$ , i.e., use  $P_{X|T}(x|D_1) = P_{x|\mu}(x|\mu_{MAP})$ , where  $\mu_{MAP} = \arg \max_{\mu} P_{\mu|T}(\mu|D_1)$ . Compare the curve with those obtained above. Can you explain the results? See "what to hand in" below.

*Solution.*

The **MAP method** treats  $\mu$  as a constant variable instead of a function of  $\mu$ , hence there is only one  $\mu$ , i.e.,  $\mu_{MAP}$ . However, when the number of data is large, the results of MAP are similar to those of PE (besides the first data set).

- (d) Repeat **a** to **c** for each of the data set  $D_i$ ,  $i = 2, 3, \dots, 4$ . Can you explain the results? See "what to hand in" below.

*Solution.*

When we examine the results closer, we can see that only for **Data 1** ( $n_{cheetah} = 75$  and  $n_{grass} = 300$ ), the difference of the results between **PE method** and **MAP method** is prominent. Other than that, the results of **PE method** and **MAP method** are similar to each other. This is due to the fact that when  $n$  is large, the result of **PE method** and **MAP method** is quite similar.

- (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.

*Solution.*

From **Figure 1** to **Figure 4**, we can see that different strategies yield distinctive results. For **PE method** and **MAP method** using *Strategy 1*, both of them perform better than using *Strategy 2*. I reckon this phenomenon is due to the way that we set up the strategies. For *Strategy 1*,  $\mu_0$  is smaller for cheetah class ( $\mu_0 = 1$ ) and larger for grass class ( $\mu_0 = 3$ ), whereas for *Strategy 2*,  $\mu_0$  is equal to half the range of amplitudes of the DCT coefficient for both classes ( $\mu_0 = 3$ ). We can tell that by comparing between *Strategy 1* and *Strategy 2*, *Strategy 1* is better because it yields lower error rate than *Strategy 2* does.

## Code for PE

```
1 clear
2 clc
3
4 % —— PE METHOD ——
5
6 % load the data
7 load("Alpha.mat"); % 1 x 9 vector
8 load("Prior_2.mat"); % Strategy 1 (W0 mu0_FG mu0_BG) %TODO:
    change
9 load("TrainingSamplesDCT_subsets_8.mat", "D4_BG", "D4_FG");
    % D1: cheetah & grass
10 load("ZigZagVec.mat") % 1 x 64 vector
11 load("cheetahMat.txt") % our image
12 load("cheetah_mask.mat") % our ideal mask
13
14 % get the size of the image
15 [m, n] = size(cheetahMat);
16
17 % create a matrix
18 m = m - 7;
19 n = n - 7;
20 totalPixels = m * n;
21
22 % # of samples
23 [n_cheetah, ~] = size(D4_FG);
24 [n_grass, ~] = size(D4_BG);
25 totalSamples = (n_cheetah + n_grass);
26
27 % calculate the priors
28 Prior_cheetah = n_cheetah / totalSamples;
29 Prior_grass = n_grass / totalSamples;
30
31 % calculate the mu_MLs
32 mu_ML_cheetah = mean(D4_FG);
33 mu_ML_grass = mean(D4_BG);
34
35 % calculate the covs
```

```

36 cov_cheetah = cov(D4_FG);
37 cov_grass    = cov(D4_BG);
38
39 % results
40 errorPE_D4P2 = zeros(1, 9); %TODO: change
41
42 % Prior 1
43 for d = 1 : 9
44     % calculate the sigma_0
45     sigma_0 = diag(alpha(d) .* W0);
46
47     % calculate the mu_ns
48     mu_n_cheetah = ((n_cheetah .* sigma_0) ./ (cov_cheetah +
        n_cheetah .* sigma_0)) .* mu_ML_cheetah + ((cov_cheetah)
        ./ (cov_cheetah + n_cheetah .* sigma_0)) .* mu0_FG;
49     mu_n_grass    = ((n_grass .* sigma_0) ./ (cov_grass +
        n_grass .* sigma_0)) .* mu_ML_grass + ((cov_grass)
        ./ (cov_grass + n_grass .* sigma_0)) .* mu0_BG;
50
51     % calculate the sigma_ns
52     sigma_n_cheetah = 1 ./ ((1 ./ cov_cheetah) + (n_cheetah ./
        cov_cheetah));
53     sigma_n_grass    = 1 ./ ((1 ./ cov_grass) + (n_grass ./
        cov_grass));
54
55     % calculate SIGMAs (sigma + sigma_n) and their inverses
56     SIGMA_cheetah = sigma_n_cheetah + cov_cheetah;
57     SIGMA_grass    = sigma_n_grass + cov_grass;
58     SIGMAInv_cheetah = inv(SIGMA_cheetah);
59     SIGMAInv_grass    = inv(SIGMA_grass);
60
61     % calculate the coefficients
62     deno_cheetah = (sqrt(((2 * pi) ^ 64) * det(SIGMA_cheetah))
        );
63     deno_grass    = (sqrt(((2 * pi) ^ 64) * det(SIGMA_grass))));
64
65     % calculate DCT2 (64D)
66     error = 0; % reset the error
67     for i = 1 : m

```



```

68     parfor j = 1 : n
69         Block = cheetahMat(i : i + 7, j : j + 7);
70         Block_DCT = dct2(Block, 8, 8);
71         V = Block_DCT(:) .';
72         X = zeros(1, 64);
73         % mapping
74         for k = 1 : 64
75             X(ZigZagVec(k)) = V(k);
76         end
77
78         % calculate the i*(x)
79         P_X_likelihood_cheetah = exp(-0.5 .* (X -
            mu_n_cheetah) * (SIGMAInv_cheetah) * (X -
            mu_n_cheetah).')) ./ deno_cheetah;
80         P_X_likelihood_grass = exp(-0.5 .* (X -
            mu_n_grass) * (SIGMAInv_grass) * (X -
            mu_n_grass).')) ./ deno_grass;
81         P_X_cheetah = P_X_likelihood_cheetah *
            Prior_cheetah;
82         P_X_grass = P_X_likelihood_grass * Prior_grass
            ;
83
84         % based on the results , decide whether the pixel
            is grass or cheetah
85         if (P_X_grass > P_X_cheetah)
86             result = 0;
87         else
88             result = 1;
89         end
90
91         % compare my result with the ideal mask
92         if (result ~= cheetah_mask(i, j))
93             error = error + 1;
94         end
95     end
96 end
97 errorPE_D4P2(1, d) = error / totalPixels;
98 end

```

## Code for ML

```
1 clear
2 clc
3
4 % —— ML METHOD ——
5
6 % load the data
7 load("Alpha.mat"); % 1 x 9 vector
8 load("TrainingSamplesDCT_subsets_8.mat", "D4_BG", "D4_FG");
   % D1: cheetah & grass
9 load("ZigZagVec.mat") % 1 x 64 vector
10 load("cheetahMat.txt") % our image
11 load("cheetah_mask.mat") % our ideal mask
12
13 % get the size of the image
14 [m, n] = size(cheetahMat);
15 m = m - 7;
16 n = n - 7;
17 totalPixels = m * n;
18 maskRes = zeros(m, n);
19
20 % # of samples
21 [n_cheetah, ~] = size(D4_FG);
22 [n_grass, ~] = size(D4_BG);
23 totalSamples = (n_cheetah + n_grass);
24
25 % calculate the priors
26 Prior_cheetah = n_cheetah / totalSamples;
27 Prior_grass = n_grass / totalSamples;
28
29 % mus and covs
30 mu_cheetah = mean(D4_FG);
31 mu_grass = mean(D4_BG);
32 cov_cheetah = cov(D4_FG);
33 cov_grass = cov(D4_BG);
34
35 % calculate SIGMAs (sigma) and their inverses
36 SIGMA_cheetah = cov_cheetah;
```

```

37 SIGMA_grass    = cov_grass;
38 SIGMAInv_cheetah = inv(SIGMA_cheetah);
39 SIGMAInv_grass  = inv(SIGMA_grass);
40
41 % calculate the coefficients
42 deno_cheetah = (sqrt(((2 * pi)^64) * det(SIGMA_cheetah)));
43 deno_grass    = (sqrt(((2 * pi)^64) * det(SIGMA_grass)));
44
45 % results
46 errorML_D4 = zeros(1, 9); %TODO: change
47
48 for d = 1 : 1
49
50     % calculate DCT2 (64D)
51     error = 0;
52     for i = 1 : m
53         parfor j = 1 : n
54             Block = cheetahMat(i : i + 7, j : j + 7);
55             Block_DCT = dct2(Block, 8, 8);
56             V = Block_DCT(:).';
57             X = zeros(1, 64);
58
59             % mapping
60             for k = 1 : 64
61                 X(ZigZagVec(k)) = V(k);
62             end
63
64             % calculate the probabilities
65             % G(X, mu_n, SIGMA)
66             P_X_likelihood_cheetah = exp(-0.5 * (X -
67                 mu_cheetah) * (SIGMAInv_cheetah) * (X -
68                 mu_cheetah).')) / deno_cheetah;
69             P_X_likelihood_grass    = exp(-0.5 * (X - mu_grass)
70                 * (SIGMAInv_grass) * (X - mu_grass).')) /
71                 deno_grass;
72             P_X_cheetah = P_X_likelihood_cheetah *
73                 Prior_cheetah;
74             P_X_grass    = P_X_likelihood_grass * Prior_grass
75                 ;

```

```

70
71         % calculate the error
72         if (P_X_grass > P_X_cheetah)
73             result = 0;
74         else
75             result = 1;
76         end
77
78         % compare my result with the ideal mask
79         if (result ~= cheetah_mask(i, j))
80             error = error + 1;
81         end
82     end
83 end
84 errorML_D4(1, d) = error / totalPixels;
85 end
86
87 for k = 2 : 9
88     errorML_D4(1, k) = errorML_D4(1, 1);
89 end
90
91 save('errorML_D4.mat', 'errorML_D4')

```

## Code for MAP

```

1  clear
2  clc
3
4  % —— MAP METHOD ——
5
6  % load the data
7  load("Alpha.mat"); % 1 x 9 vector
8  load("Prior_2.mat"); % Strategy 1 (W0 mu0_FG mu0_BG)
9  load("TrainingSamplesDCT_subsets_8.mat", "D1_BG", "D1_FG");
   % D1: cheetah & grass
10 load("ZigZagVec.mat") % 1 x 64 vector
11 load("cheetahMat.txt") % our image
12 load("cheetah_mask.mat") % our ideal mask
13

```

```

14 % get the size of the image
15 [m, n] = size(cheetahMat);
16 m = m - 7;
17 n = n - 7;
18 maskRes = zeros(m, n);
19 totalPixels = m * n;
20
21 % # of samples
22 [n_cheetah, ~] = size(D1_FG);
23 [n_grass, ~] = size(D1_BG);
24 totalSamples = (n_cheetah + n_grass);
25
26 % calculate the priors
27 Prior_cheetah = n_cheetah / totalSamples;
28 Prior_grass = n_grass / totalSamples;
29
30 % mu_MLs and covs
31 mu_ML_cheetah = mean(D1_FG);
32 mu_ML_grass = mean(D1_BG);
33 cov_cheetah = cov(D1_FG);
34 cov_grass = cov(D1_BG);
35
36 % calculate the sigma_0
37 d = 6; % which alpha am I using
38 sigma_0 = diag(alpha(d) .* W0);
39
40 % calculate the mu_ns
41 mu_n_cheetah = ((n_cheetah .* sigma_0) ./ (cov_cheetah +
    n_cheetah .* sigma_0)) .* mu_ML_cheetah + ((cov_cheetah) ./
    (cov_cheetah + n_cheetah .* sigma_0)) .* mu0_FG;
42 mu_n_grass = ((n_grass .* sigma_0) ./ (cov_grass +
    n_grass .* sigma_0)) .* mu_ML_grass + ((cov_grass) ./
    (cov_grass + n_grass .* sigma_0)) .* mu0_BG;
43
44 % calculate SIGMAs (sigma) and their inverses
45 SIGMA_cheetah = cov_cheetah;
46 SIGMA_grass = cov_grass;
47 SIGMAInv_cheetah = inv(SIGMA_cheetah);
48 SIGMAInv_grass = inv(SIGMA_grass);

```

```

49
50 % calculate the coefficients
51 deno_cheetah = (sqrt(((2 * pi)^64) * det(SIGMA_cheetah)));
52 deno_grass    = (sqrt(((2 * pi)^64) * det(SIGMA_grass)));
53
54 % calculate DCT2 (64D)
55 error = 0;
56 for i = 1 : m
57     for j = 1 : n
58         Block = cheetahMat(i : i + 7, j : j + 7);
59         Block_DCT = dct2(Block, 8, 8);
60         V = Block_DCT(:) .';
61         X = zeros(1, 64);
62
63         % mapping
64         for k = 1 : 64
65             X(ZigZagVec(k)) = V(k);
66         end
67
68         % calculate the probabilities
69         % G(X, mu_n, SIGMA)
70         P_X_likelihood_cheetah = exp(-0.5 * (X - mu_n_cheetah)
71             * (SIGMAInv_cheetah) * (X - mu_n_cheetah) .') /
72             deno_cheetah;
71         P_X_likelihood_grass    = exp(-0.5 * (X - mu_n_grass) *
72             (SIGMAInv_grass) * (X - mu_n_grass) .') / deno_grass
73             ;
72         P_X_cheetah = P_X_likelihood_cheetah * Prior_cheetah;
73         P_X_grass    = P_X_likelihood_grass    * Prior_grass;
74
75         % based on the results , decide whether the pixel is
76             grass or cheetah
76         if (P_X_grass > P_X_cheetah)
77             maskRes(i, j) = 0;
78         else
79             maskRes(i, j) = 1;
80         end
81
82         % calculate the error

```

```

83         if (P_X_grass > P_X_cheetah)
84             result = 0;
85         else
86             result = 1;
87         end
88
89         % calculate the error rate
90         if(result ~= cheetah_mask(i, j))
91             error = error + 1;
92         end
93
94     end
95 end
96
97 % display my mask
98 imshow(maskRes)
99 title("MAP Method")
100
101 % result (error)
102 errorRate = error / totalPixels;
103 disp(errorRate)

```

## Code for Plotting Graphs

```

1  clear
2  clc
3  clf
4
5  % —— Plot Graphs ——
6
7  % load results
8  load("Alpha.mat")
9  load("errorMAP_D4P2.mat")
10 load("errorPE_D4P2.mat")
11 load("errorML_D4.mat")
12
13 % set up x-axis and y-axes
14 x = alpha;
15

```

```

16 y1 = errorBayes_D4P2;
17 y2 = errorML_D4;
18 y3 = errorMAP_D4P2;
19
20 % plot the graph
21 semilogx(x, y1, x, y2, x, y3)
22 title("Data 4 Strategy 2")
23 legend('PE', 'ML', 'MAP')
24 grid on

```