ECE 271A Homework Set Three (Quiz)

CHENG-YAN JUANG

1) Predictive distribution

Because the Gaussian prior is conjugate to the Gaussian likelihood, the posterior of the mean remains Gaussian, the predictive distribution is:

$$P_{X|T}(\, x \mid D\,) = G(x, \mu_n, \Sigma + \Sigma_0)$$

Where:

$$\mu_n = \frac{n\Sigma_0}{\Sigma + n\Sigma_0}\mu_{ML} + \frac{\Sigma}{\Sigma + n\Sigma_0}\mu_0$$

$$\Sigma_n = \Sigma + \Sigma_0 = \frac{\Sigma\Sigma_0}{\Sigma + n\Sigma_0}$$

2) MAP

The MAP classifier is:

$$P_{X|T}(\, x \mid D\,) = G(x, \mu_n, \Sigma)$$

3) ML

The ML classifier is:

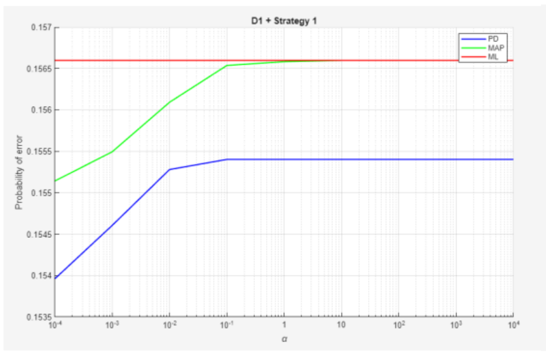
$$\Sigma_{ML} = \frac{1}{N}\sum_{i=1}^{n}(x^i - \mu_{ML})(x^i - \mu_{ML})^T$$

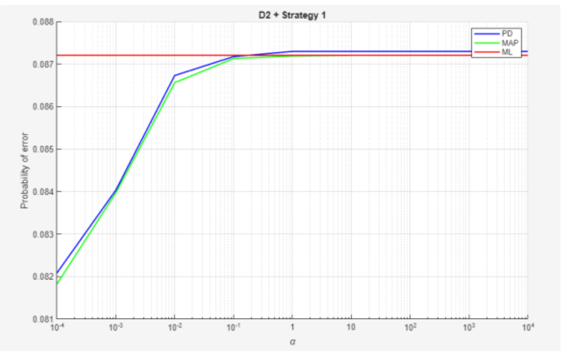
Where:

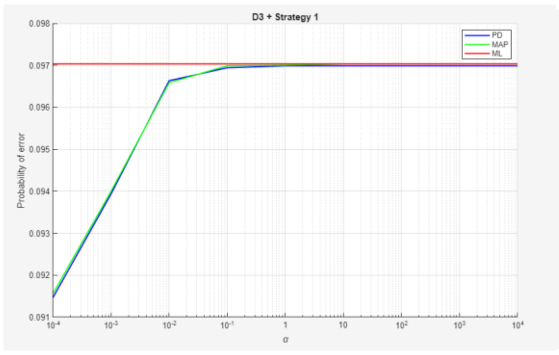
$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{n} x^i$$

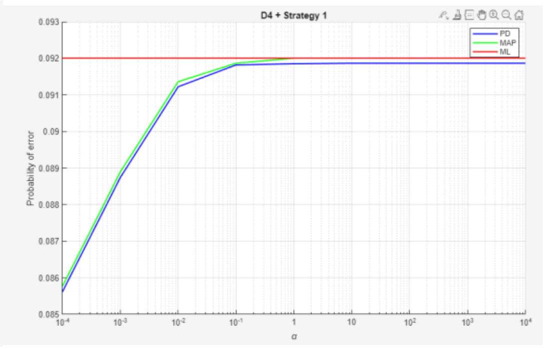
4) Figures

- <u>Strategy 1:</u>

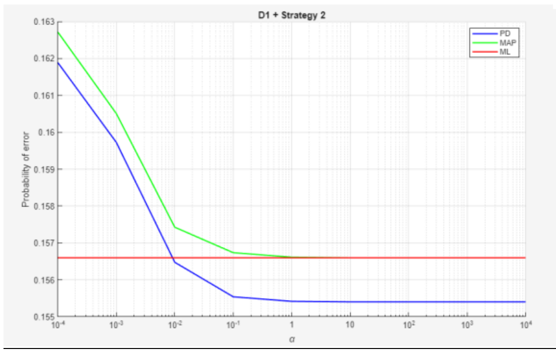


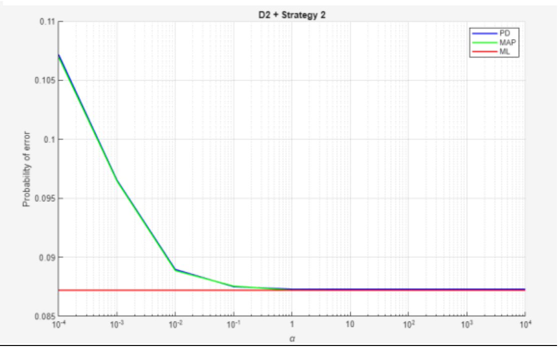


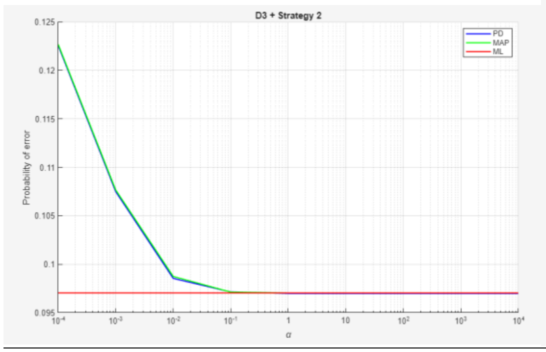


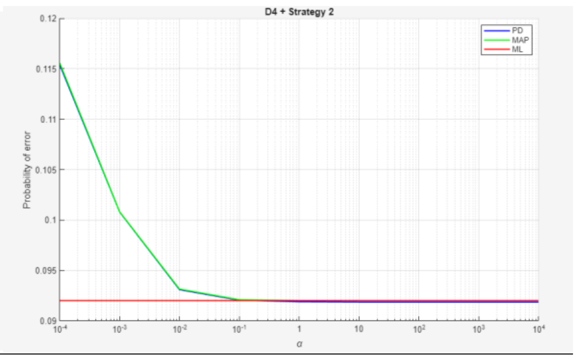


- <u>Strategy 2:</u>









5) Analysis

- Relative Behavior of PD、MAP、ML in Dataset:

The ML curve is a horizontal line because the Maximum Likelihood estimate relies solely on the training data and is independent of alpha. For PD and MAP, as alpha increases (higher uncertainty in the prior), both curves converge to the ML error rate when there is enough data. This convergence occurs because, as alpha increases, the prior variance Σ_0 also increases. Consequently, the posterior mean $\mu_n$becomes dominated by the data term μ_{ML}, making the Bayesian solutions identical to the ML solution. In Strategy 1, PD has the lowest error rate, often slightly better than MAP. This is because PD accounts for the uncertainty in the parameter estimation, whereas MAP uses only the sample covariance and ignores estimation uncertainty. This makes PD more robust.

- Relative Behavior for different Dataset D1 to D4:

As we move from datasets D1 (smallest) to D4 (largest), the error rates for all methods generally decrease, as more training data leads to more accurate estimation of the class-conditional densities, and in larger datasets, PD & MAP converge to the ML line faster. The posterior mean is a weighted average of the prior mean μ_0 and the sample mean μ_{ML}. The weight of the likelihood is proportional to the sample size N. For large N, the data evidence overwhelms the prior belief faster; thus, the prior loses its influence quickly, and the solution behaves like ML even for relatively small alpha. Conversely, for smaller N, the prior remains influential for a wider range of alpha.

- Strategy 1 vs. Strategy 2:

Strategy 1 uses distinct, class-specific means derived from previous knowledge. This good prior information helps guide the classification, especially when the training data is scarce and noisy. On the other hand, Strategy 2 is a non-informative, poor prior that uses a generic mean that is identical for both classes. When alpha is small, we force the model to believe that Cheetahs and Grass look the same. This poor prior misleads the classifier. Only when alpha increases (we discard this bad prior) does the performance recover to the ML level.

This comparison highlights that Bayesian methods are powerful but is sensitive to the quality of the prior. A good prior (Strategy 1) improves performance (acts as regularization), while a bad prior (Strategy 2) degrades it. However, with enough uncertainty (alpha) or enough data, the data eventually corrects the bad prior, like the professor always says: “Data always wins”.

Table of Contents

```
clear; clc;

load('TrainingSamplesDCT_subsets_8.mat');
load('Alpha.mat');
test_img = im2double(imread('cheetah.bmp'));
mask     = im2double(imread('cheetah_mask.bmp'));
alpha_list = alpha(:);
```

4) zig-zag

```
zig_zag_array = load('Zig-Zag Pattern.txt');
zig_zag_array = double(zig_zag_array) + 1;
order_coef = zeros(1, 64);
for r = 1:8
    for c = 1:8
        k = zig_zag_array(r,c);
        order_coef(k) = sub2ind([8,8], r, c);
    end
end

[H,W] = size(test_img);

GT = mask(1:(H-7), 1:(W-7)) > 0.5;

cheetah_data   = {D1_FG, D2_FG, D3_FG, D4_FG};
grass_data   = {D1_BG, D2_BG, D3_BG, D4_BG};
data_set  = {'D1','D2','D3','D4'};

prior_files = {'Prior_1.mat','Prior_2.mat'};
prior_names = {'Strategy 1','Strategy 2'};
```

Strategy 1 / Strategy 2

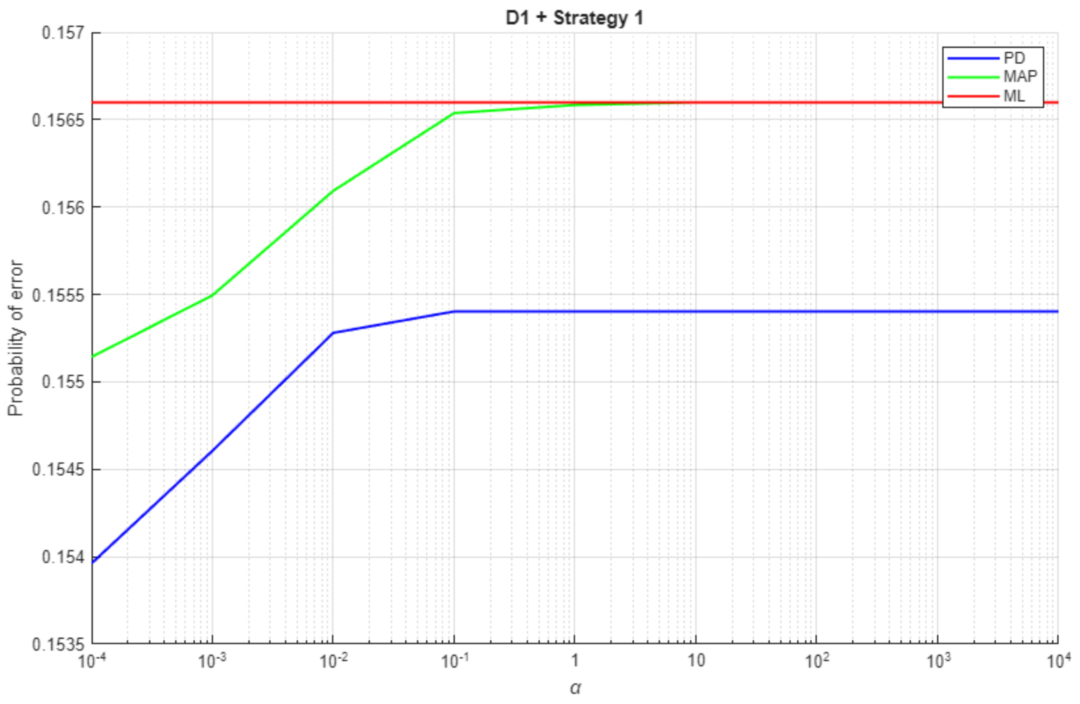
```
for strat = 1:2

    load(prior_files{strat});

    mu0_FG = mu0_FG(:);
    mu0_BG = mu0_BG(:);
    w0     = W0(:);

    fprintf('\n===== %s =====\n', prior_names{strat});

    for ds = 1:4

        fprintf("\n< Running %s + %s >\n", data_set{ds}, prior_names{strat});

        TrainsampleDCT_FG = cheetah_data{ds};
        TrainsampleDCT_BG = grass_data{ds};

        N_FG = size(TrainsampleDCT_FG, 1);
        N_BG = size(TrainsampleDCT_BG, 1);

        Py_cheetah = N_FG / (N_FG + N_BG);
        Py_grass   = 1 - Py_cheetah;

        mu_FG = mean(TrainsampleDCT_FG, 1)';
        mu_BG = mean(TrainsampleDCT_BG, 1)';
        cov_FG = cov(TrainsampleDCT_FG, 1);
        cov_BG = cov(TrainsampleDCT_BG, 1);
```

```
< Running D1 + Strategy 1 >


< Running D2 + Strategy 1 >


< Running D3 + Strategy 1 >


< Running D4 + Strategy 1 >


< Running D1 + Strategy 2 >


< Running D2 + Strategy 2 >


< Running D3 + Strategy 2 >


< Running D4 + Strategy 2 >
```

(a) Predictive Distribution

```
        PD_error = zeros(length(alpha_list),1);

        for a = 1:length(alpha_list)

            cov0 = diag(alpha_list(a) * w0);
            % BG posterior
            W1BG = (N_BG * cov0) / (cov_BG + N_BG*cov0);
            W2BG =  cov_BG         / (cov_BG + N_BG*cov0);
            muN_BG = W1BG*mu_BG + W2BG*mu0_BG;
            SigmaN_BG = (cov_BG * cov0) / (cov_BG + N_BG*cov0);
            SigmaN_BG_PD = ((cov_BG + SigmaN_BG) + (cov_BG + SigmaN_BG)') /
2;

            % FG posterior
            W1FG = (N_FG * cov0) / (cov_FG + N_FG*cov0);
            W2FG =  cov_FG         / (cov_FG + N_FG*cov0);
            muN_FG = W1FG*mu_FG + W2FG*mu0_FG;
            SigmaN_FG = (cov_FG * cov0) / (cov_FG + N_FG*cov0);
            SigmaN_FG_PD = ((cov_FG + SigmaN_FG) + (cov_FG + SigmaN_FG)') /
2;

            % Classification
            mask_pred = zeros(H-7, W-7);
            for i = 1:(H-7)
                for j = 1:(W-7)
                    blk = test_img(i:i+7, j:j+7);
                    D   = dct2(blk);
                    x64 = D(order_coef).';

                    ll_BG = log_gauss(x64, muN_BG, SigmaN_BG_PD) +
log(Py_grass);
                    ll_FG = log_gauss(x64, muN_FG, SigmaN_FG_PD) +
log(Py_cheetah);

                    mask_pred(i,j) = (ll_FG > ll_BG);
                end
            end

            PD_error(a) = mean(mask_pred(:) ~= GT(:));
            fprintf(" (a) Alpha=%g, PD error=%.4f\n", alpha_list(a),
PD_error(a));
        end
```

```
 (a) Alpha=0.0001, PD error=0.0821
 (a) Alpha=0.001, PD error=0.0840
 (a) Alpha=0.01, PD error=0.0867
 (a) Alpha=0.1, PD error=0.0872
 (a) Alpha=1, PD error=0.0873
 (a) Alpha=10, PD error=0.0873
 (a) Alpha=100, PD error=0.0873
 (a) Alpha=1000, PD error=0.0873
 (a) Alpha=10000, PD error=0.0873
```



```
(a) Alpha=0.0001, PD error=0.0915
(a) Alpha=0.001, PD error=0.0939
(a) Alpha=0.01, PD error=0.0966
(a) Alpha=0.1, PD error=0.0969
(a) Alpha=1, PD error=0.0970
(a) Alpha=10, PD error=0.0970
(a) Alpha=100, PD error=0.0970
(a) Alpha=1000, PD error=0.0970
(a) Alpha=10000, PD error=0.0970

(a) Alpha=0.0001, PD error=0.0856
(a) Alpha=0.001, PD error=0.0887
(a) Alpha=0.01, PD error=0.0912
(a) Alpha=0.1, PD error=0.0918
(a) Alpha=1, PD error=0.0919
(a) Alpha=10, PD error=0.0919
(a) Alpha=100, PD error=0.0919
(a) Alpha=1000, PD error=0.0919
(a) Alpha=10000, PD error=0.0919

(a) Alpha=0.0001, PD error=0.1619
(a) Alpha=0.001, PD error=0.1597
(a) Alpha=0.01, PD error=0.1565
(a) Alpha=0.1, PD error=0.1555
(a) Alpha=1, PD error=0.1554
(a) Alpha=10, PD error=0.1554
(a) Alpha=100, PD error=0.1554
(a) Alpha=1000, PD error=0.1554
(a) Alpha=10000, PD error=0.1554
```

```
(a) Alpha=0.0001, PD error=0.1072
(a) Alpha=0.001, PD error=0.0965
(a) Alpha=0.01, PD error=0.0890
(a) Alpha=0.1, PD error=0.0875
(a) Alpha=1, PD error=0.0873
(a) Alpha=10, PD error=0.0873
(a) Alpha=100, PD error=0.0873
(a) Alpha=1000, PD error=0.0873
(a) Alpha=10000, PD error=0.0873
```

```
(a) Alpha=0.0001, PD error=0.1226
(a) Alpha=0.001, PD error=0.1075
(a) Alpha=0.01, PD error=0.0985
(a) Alpha=0.1, PD error=0.0971
(a) Alpha=1, PD error=0.0970
(a) Alpha=10, PD error=0.0970
(a) Alpha=100, PD error=0.0970
(a) Alpha=1000, PD error=0.0970
(a) Alpha=10000, PD error=0.0970
```

```
(a) Alpha=0.0001, PD error=0.1154
(a) Alpha=0.001, PD error=0.1008
(a) Alpha=0.01, PD error=0.0931
(a) Alpha=0.1, PD error=0.0921
(a) Alpha=1, PD error=0.0919
(a) Alpha=10, PD error=0.0919
(a) Alpha=100, PD error=0.0919
(a) Alpha=1000, PD error=0.0919
(a) Alpha=10000, PD error=0.0919
```

(b) ML

```
mask_ML = zeros(H-7, W-7);
for i = 1:(H-7)
    for j = 1:(W-7)
        blk = test_img(i:i+7, j:j+7);
        D   = dct2(blk);
        x64 = D(order_coef).';

        ll_BG = log_gauss(x64, mu_BG, cov_BG) + log(Py_grass);
        ll_FG = log_gauss(x64, mu_FG, cov_FG) + log(Py_cheetah);
        mask_ML(i,j) = (ll_FG > ll_BG);
    end
end
ML_error = mean(mask_ML(:) ~= GT(:));
ML_error_vec = ML_error * ones(size(alpha_list));
fprintf(" (b) ML error = %.4f\n", ML_error);
```

```
(b) ML error = 0.1566
```

```
(b) ML error = 0.0872
```

```
(b) ML error = 0.0970
```

```
(b) ML error = 0.0920
```

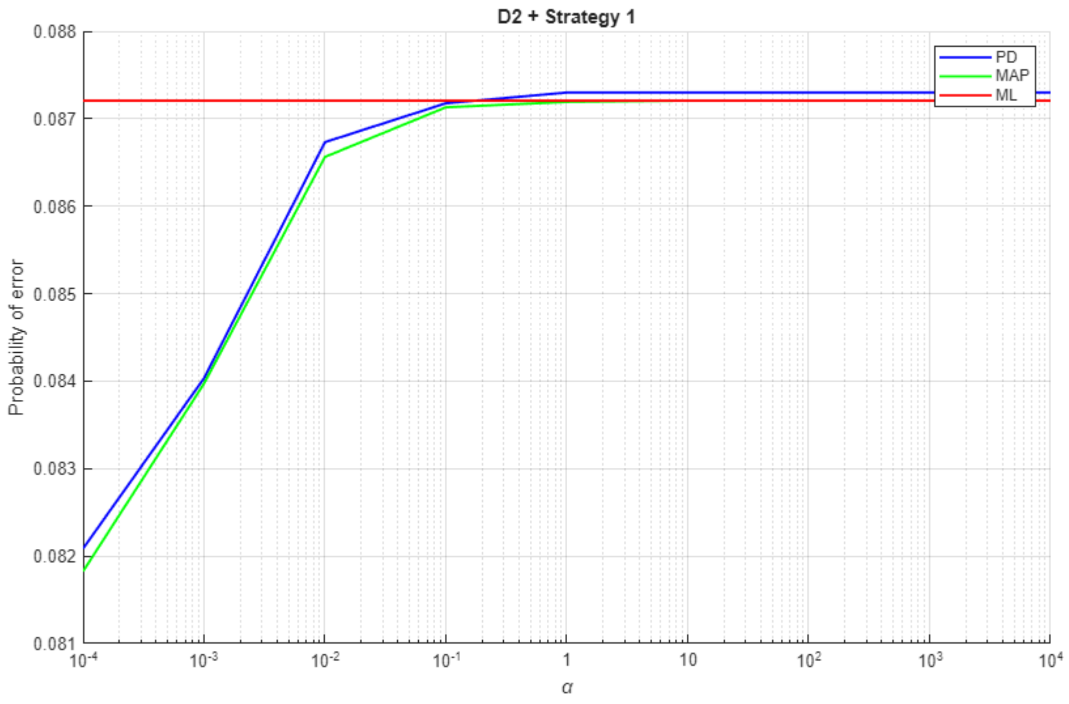
```
(b) ML error = 0.1566

(b) ML error = 0.0872

(b) ML error = 0.0970

(b) ML error = 0.0920
```

(c) MAP

```
        MAP_error = zeros(length(alpha_list),1);

        for a = 1:length(alpha_list)

            cov0 = diag(alpha_list(a) * w0);

            W1BG = (N_BG * cov0) / (cov_BG + N_BG*cov0);
            W2BG =  cov_BG          / (cov_BG + N_BG*cov0);
            muMAP_BG = W1BG*mu_BG + W2BG*mu0_BG;

            W1FG = (N_FG * cov0) / (cov_FG + N_FG*cov0);
            W2FG =  cov_FG          / (cov_FG + N_FG*cov0);
            muMAP_FG = W1FG*mu_FG + W2FG*mu0_FG;

            mask_MAP = zeros(H-7, W-7);
            for i = 1:(H-7)
                for j = 1:(W-7)
                    blk = test_img(i:i+7, j:j+7);
                    D   = dct2(blk);
                    x64 = D(order_coef).';

                    ll_BG = log_gauss(x64, muMAP_BG, cov_BG) + log(Py_grass);
                    ll_FG = log_gauss(x64, muMAP_FG, cov_FG) +
log(Py_cheetah);

                    mask_MAP(i,j) = (ll_FG > ll_BG);
                end
            end

            MAP_error(a) = mean(mask_MAP(:) ~= GT(:));
            fprintf(" (c) Alpha=%g, MAP error=%.4f\n", alpha_list(a),
MAP_error(a));
        end
```

```
(c) Alpha=0.0001, MAP error=0.1551
(c) Alpha=0.001, MAP error=0.1555
(c) Alpha=0.01, MAP error=0.1561
(c) Alpha=0.1, MAP error=0.1565
(c) Alpha=1, MAP error=0.1566
(c) Alpha=10, MAP error=0.1566
(c) Alpha=100, MAP error=0.1566
(c) Alpha=1000, MAP error=0.1566
(c) Alpha=10000, MAP error=0.1566
```

```
(c) Alpha=0.0001, MAP error=0.0818
(c) Alpha=0.001, MAP error=0.0840
(c) Alpha=0.01, MAP error=0.0866
(c) Alpha=0.1, MAP error=0.0871
(c) Alpha=1, MAP error=0.0872
(c) Alpha=10, MAP error=0.0872
(c) Alpha=100, MAP error=0.0872
(c) Alpha=1000, MAP error=0.0872
(c) Alpha=10000, MAP error=0.0872

(c) Alpha=0.0001, MAP error=0.0916
(c) Alpha=0.001, MAP error=0.0940
(c) Alpha=0.01, MAP error=0.0966
(c) Alpha=0.1, MAP error=0.0970
(c) Alpha=1, MAP error=0.0970
(c) Alpha=10, MAP error=0.0970
(c) Alpha=100, MAP error=0.0970
(c) Alpha=1000, MAP error=0.0970
(c) Alpha=10000, MAP error=0.0970
```



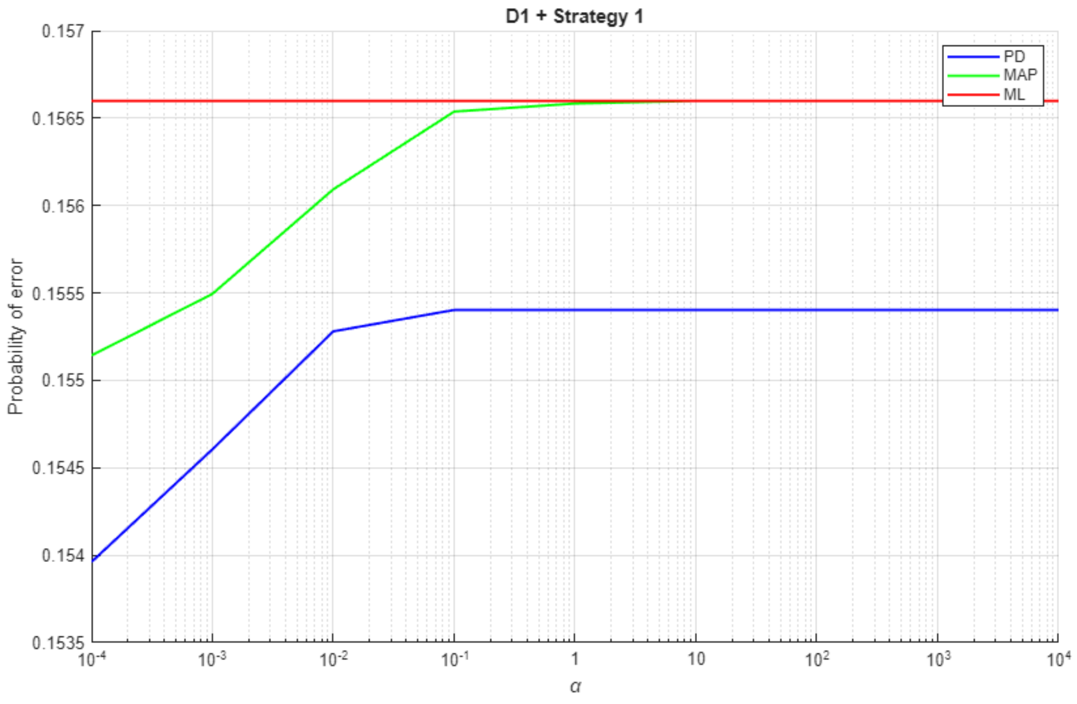
```
(c) Alpha=0.0001, MAP error=0.0858
(c) Alpha=0.001, MAP error=0.0889
(c) Alpha=0.01, MAP error=0.0914
(c) Alpha=0.1, MAP error=0.0919
(c) Alpha=1, MAP error=0.0920
(c) Alpha=10, MAP error=0.0920
(c) Alpha=100, MAP error=0.0920
(c) Alpha=1000, MAP error=0.0920
(c) Alpha=10000, MAP error=0.0920
```

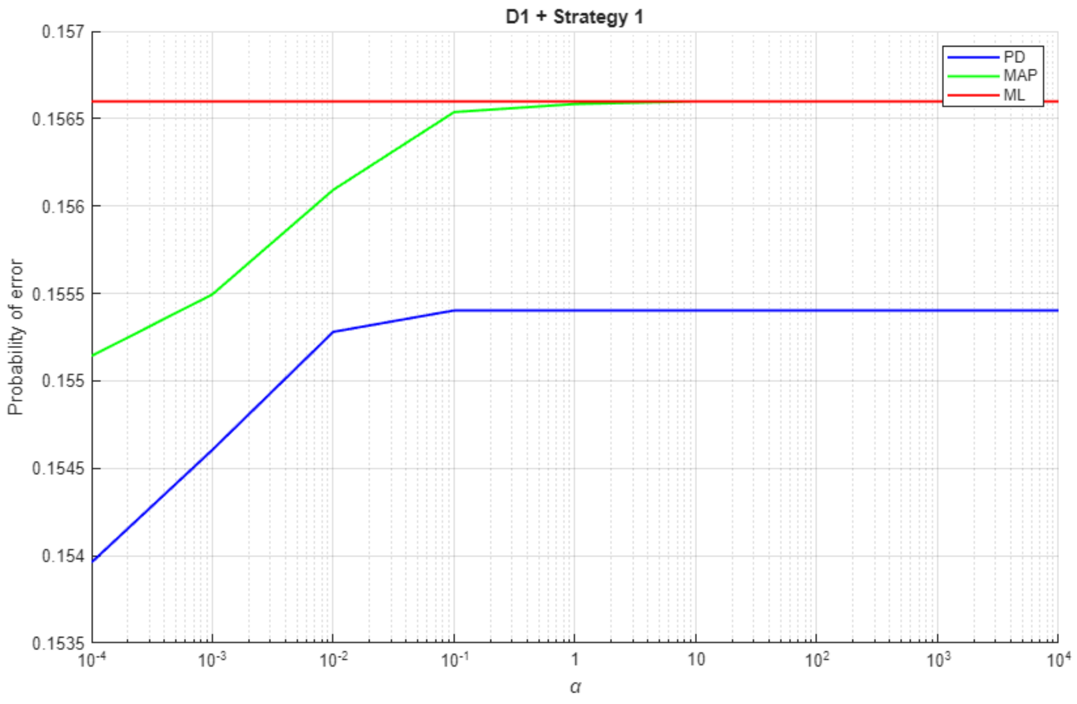
```
(c) Alpha=0.0001, MAP error=0.1627
(c) Alpha=0.001, MAP error=0.1605
(c) Alpha=0.01, MAP error=0.1574
(c) Alpha=0.1, MAP error=0.1567
(c) Alpha=1, MAP error=0.1566
(c) Alpha=10, MAP error=0.1566
(c) Alpha=100, MAP error=0.1566
(c) Alpha=1000, MAP error=0.1566
(c) Alpha=10000, MAP error=0.1566

(c) Alpha=0.0001, MAP error=0.1070
(c) Alpha=0.001, MAP error=0.0965
(c) Alpha=0.01, MAP error=0.0889
(c) Alpha=0.1, MAP error=0.0876
(c) Alpha=1, MAP error=0.0872
(c) Alpha=10, MAP error=0.0872
(c) Alpha=100, MAP error=0.0872
(c) Alpha=1000, MAP error=0.0872
(c) Alpha=10000, MAP error=0.0872

(c) Alpha=0.0001, MAP error=0.1227
(c) Alpha=0.001, MAP error=0.1077
(c) Alpha=0.01, MAP error=0.0987
(c) Alpha=0.1, MAP error=0.0971
(c) Alpha=1, MAP error=0.0970
(c) Alpha=10, MAP error=0.0970
(c) Alpha=100, MAP error=0.0970
(c) Alpha=1000, MAP error=0.0970
(c) Alpha=10000, MAP error=0.0970

(c) Alpha=0.0001, MAP error=0.1156
(c) Alpha=0.001, MAP error=0.1008
(c) Alpha=0.01, MAP error=0.0932
(c) Alpha=0.1, MAP error=0.0921
(c) Alpha=1, MAP error=0.0920
(c) Alpha=10, MAP error=0.0920
(c) Alpha=100, MAP error=0.0920
(c) Alpha=1000, MAP error=0.0920
(c) Alpha=10000, MAP error=0.0920
```

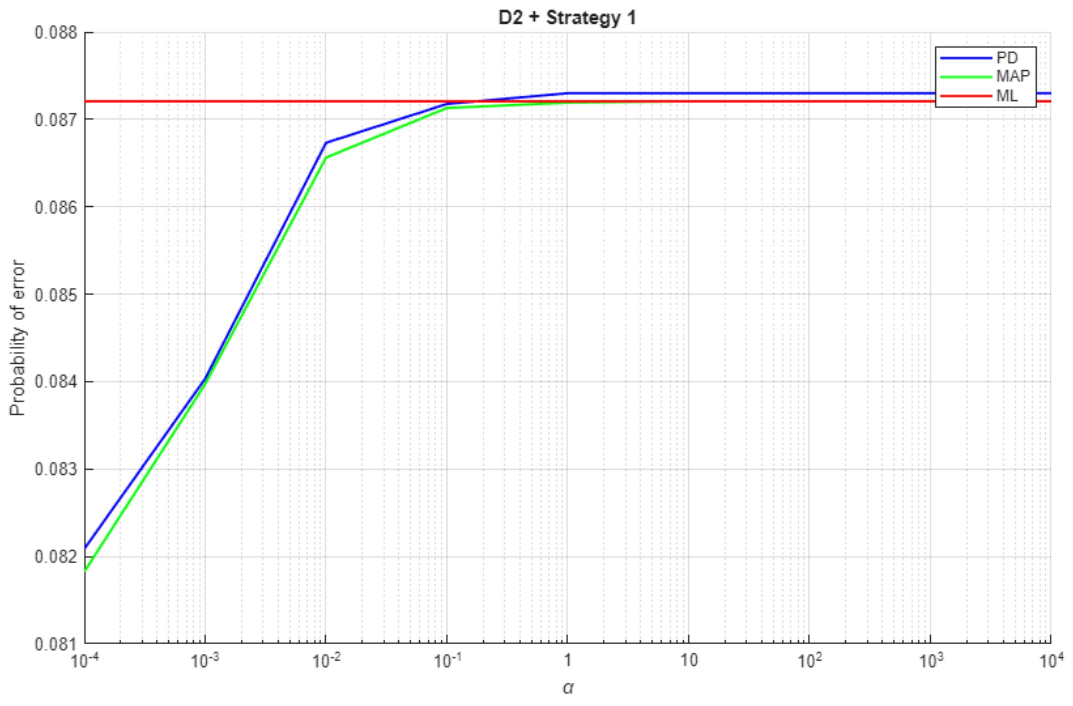
Predictive distribution + ML + MAP plot

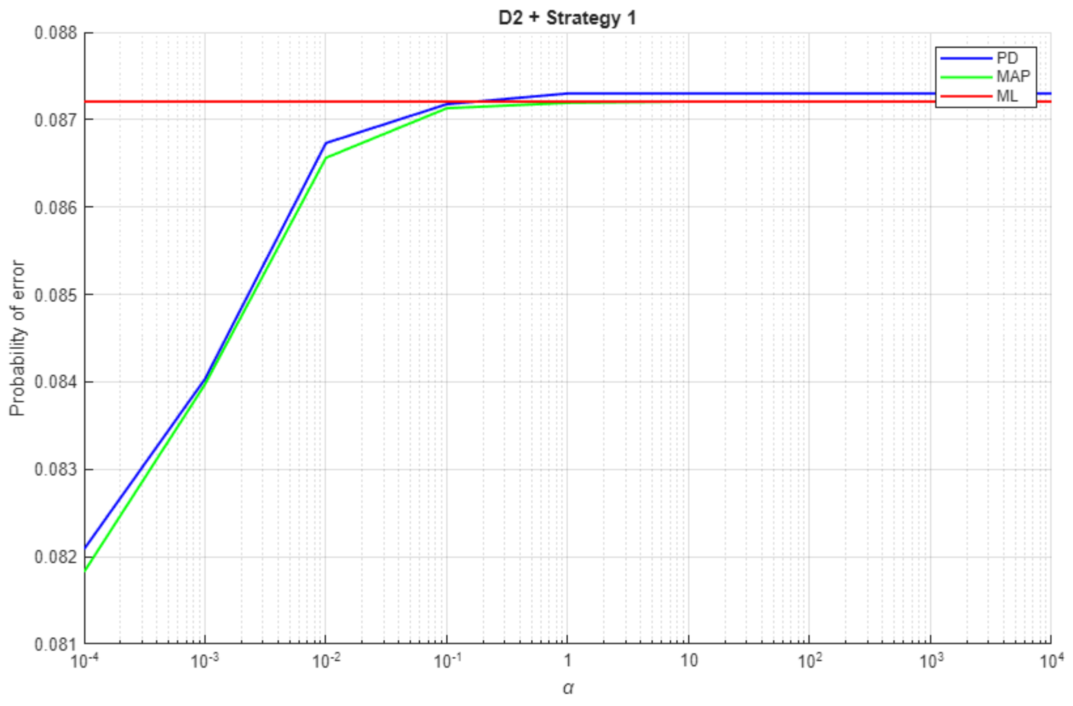
```
figure; hold on; grid on;
semilogx(alpha_list, PD_error, 'b', 'LineWidth', 1.4);
semilogx(alpha_list, MAP_error, 'g', 'LineWidth', 1.4);
semilogx(alpha_list, ML_error_vec, 'r', 'LineWidth', 1.4);

set(gca, 'XScale', 'log');
set(gca, 'XTick', alpha_list);
set(gca, 'XTickLabel', {'10^{-4}','10^{-3}','10^{-2}','10^{-1}', ...
                        '1','10','10^{2}','10^{3}','10^{4}'});

xlabel('\alpha'); ylabel('Probability of error');
```

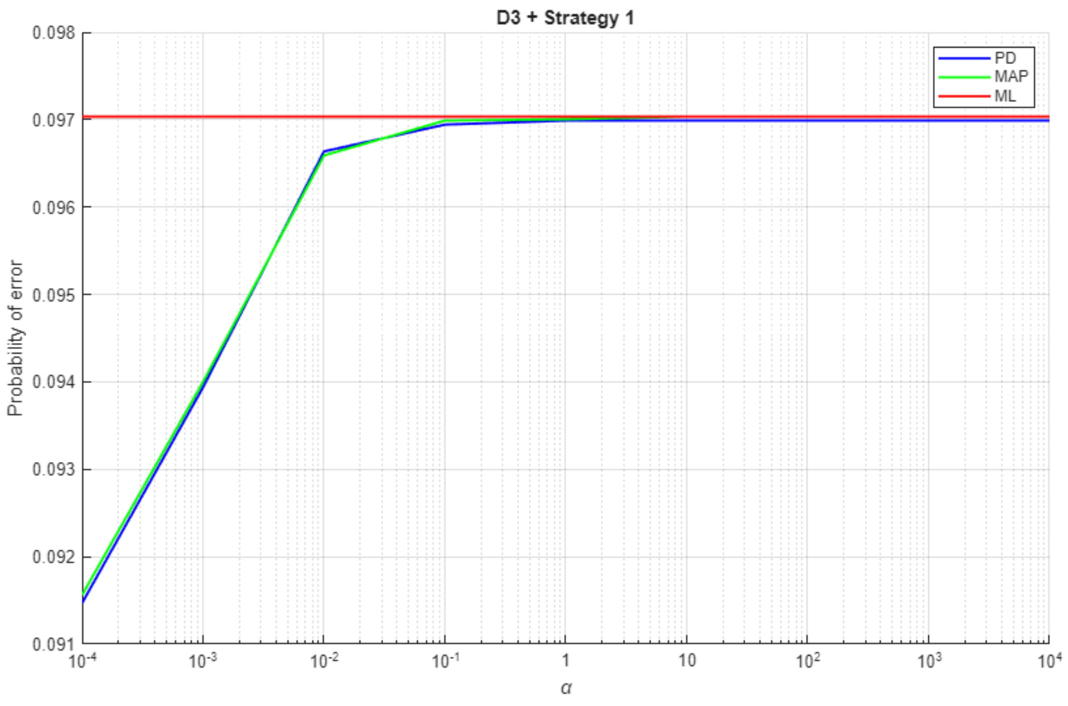
```
title(sprintf('%s + %s', data_set{ds}, prior_names{strat}));
legend('PD','MAP','ML');
```

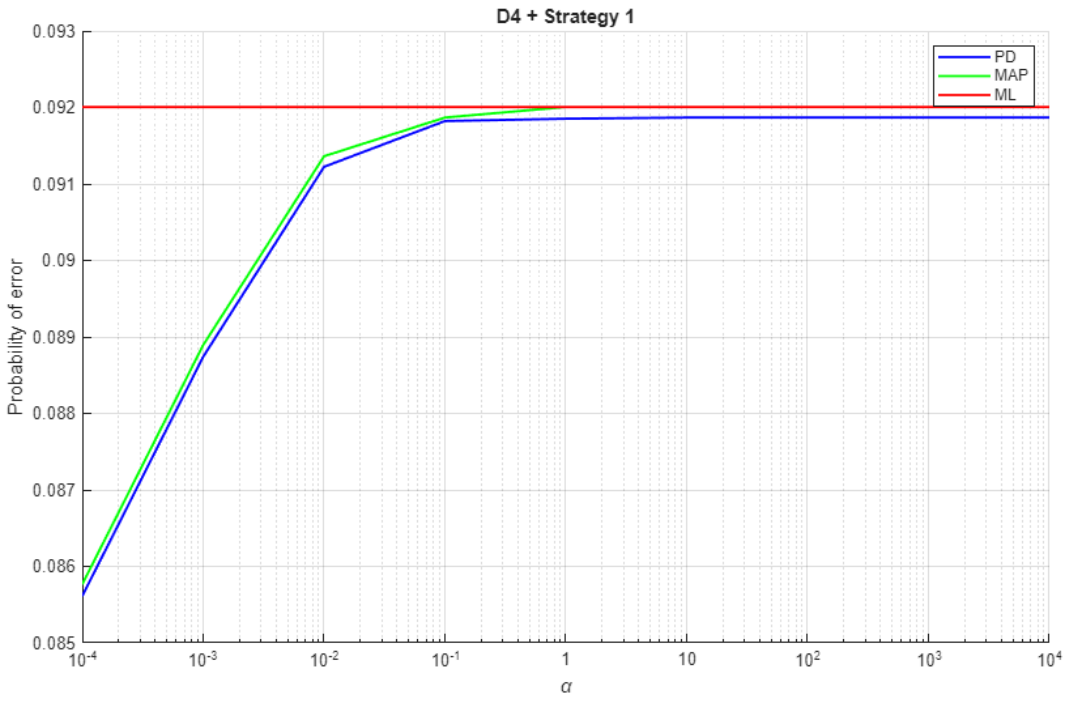


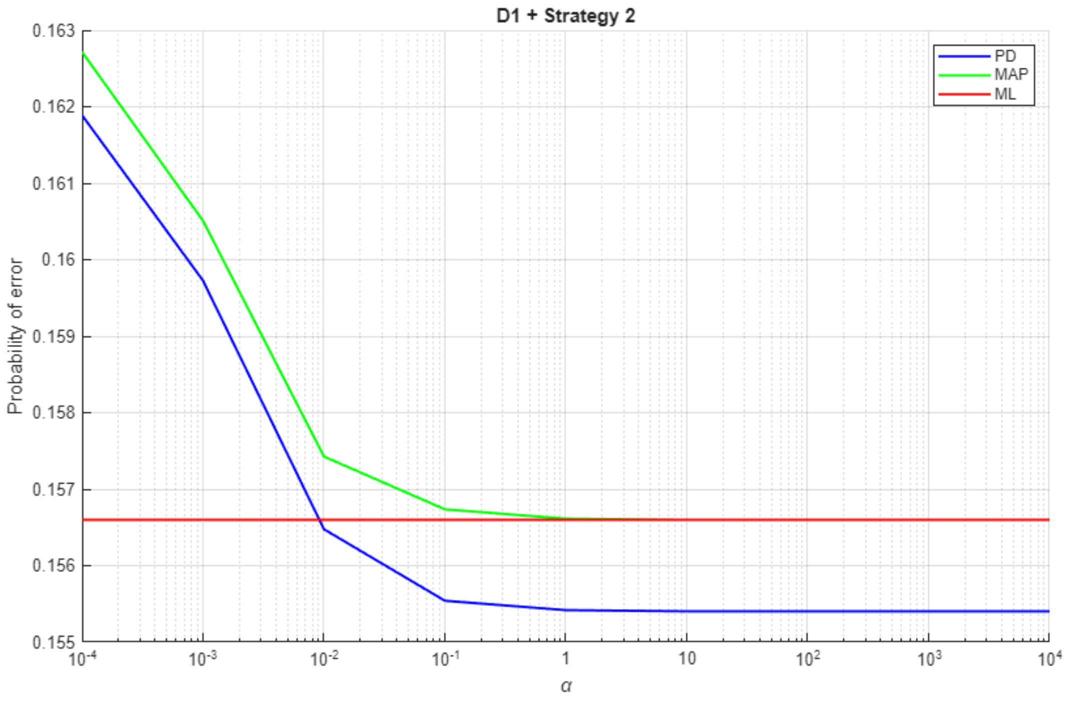


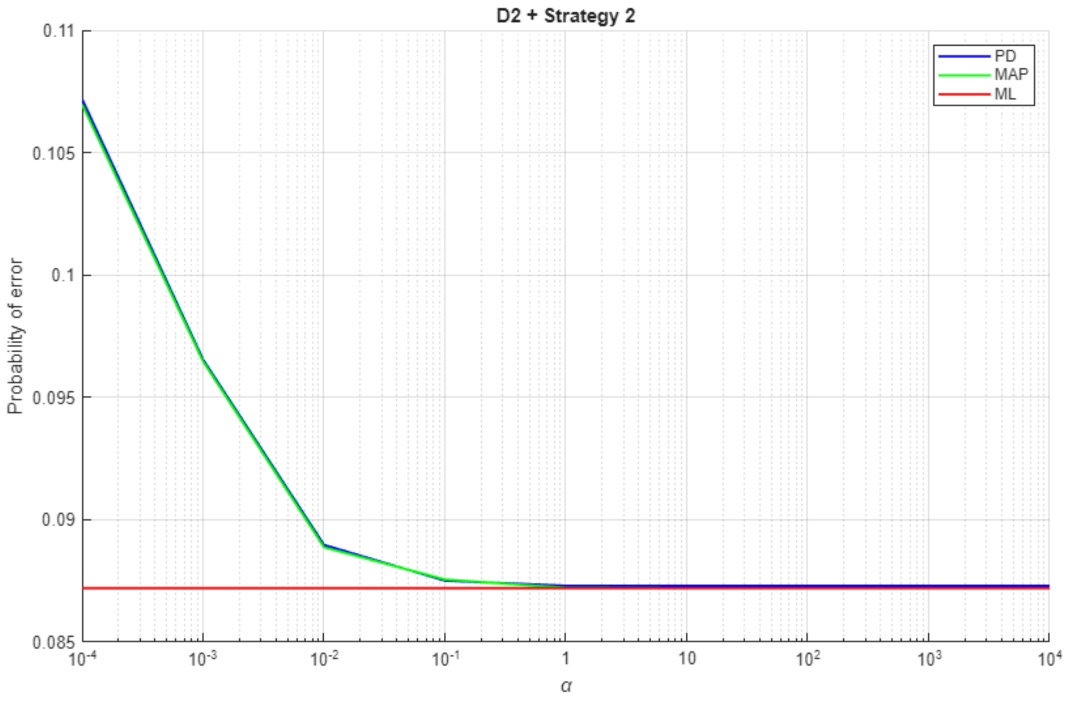

D2 + Strategy 1
Probability of error
0.088
0.087
0.086
0.085
0.084
0.083
0.082
0.081
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α
PD
MAP
ML

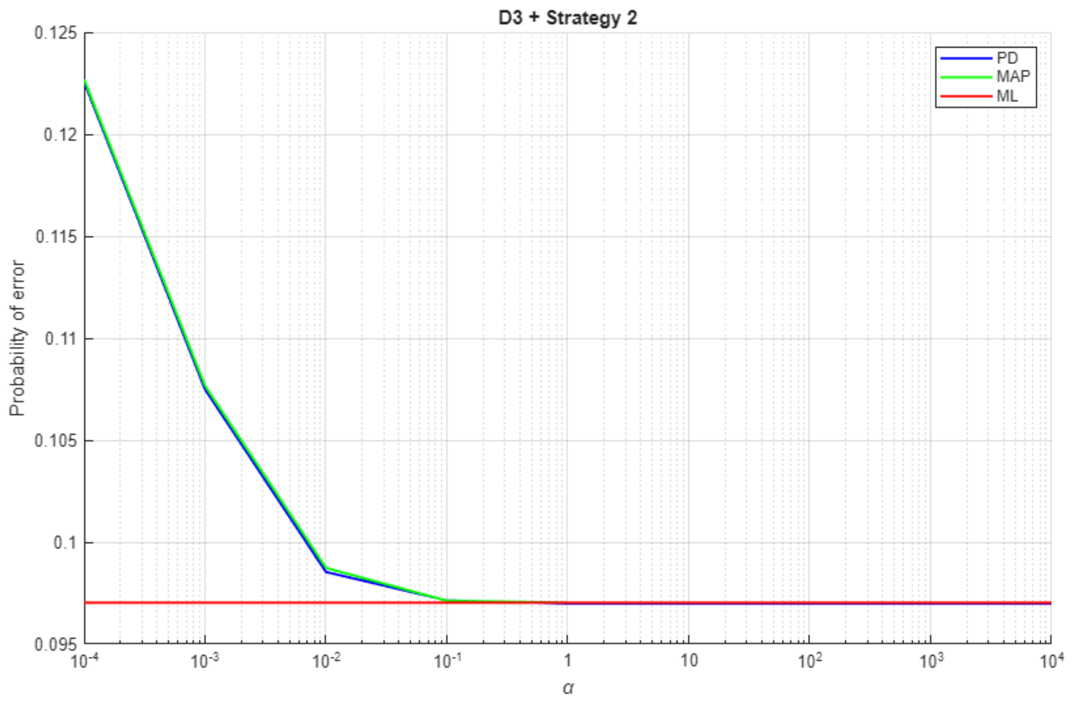

D2 + Strategy 1
Probability of error
0.088
0.087
0.086
0.085
0.084
0.083
0.082
0.081
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α
PD
MAP
ML

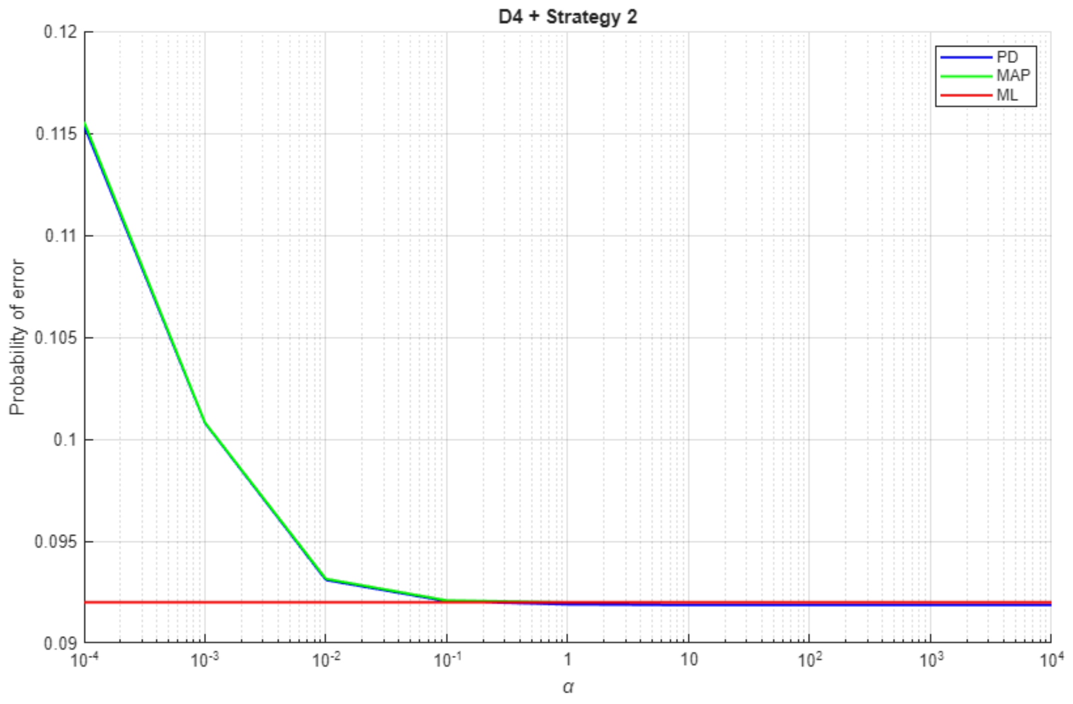

D3 + Strategy 1
PD
MAP
ML
Probability of error
0.098
0.097
0.096
0.095
0.094
0.093
0.092
0.091
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α


D4 + Strategy 1
PD
MAP
ML
Probability of error
0.093
0.092
0.091
0.09
0.089
0.088
0.087
0.086
0.085
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α


D1 + Strategy 2
PD
MAP
ML
Probability of error
0.163
0.162
0.161
0.16
0.159
0.158
0.157
0.156
0.155
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α


D2 + Strategy 2
PD
MAP
ML
Probability of error
0.11
0.105
0.1
0.095
0.09
0.085
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
α





```
    end
end


===== Strategy 1 =====
```

log function

```
function log_p = log_gauss(x, mu, Sigma)
    x  = x(:);
    mu = mu(:);

    diff = x - mu;
    % (x - mu)' * inv(Sigma) * (x - mu)
    quad = diff' * (Sigma \ diff);
    ld = log(det(Sigma));

    d = length(x);
    log_p = -0.5 * (quad + ld + d*log(2*pi));
end
```

```
 (a) Alpha=0.0001, PD error=0.1540
 (a) Alpha=0.001, PD error=0.1546
 (a) Alpha=0.01, PD error=0.1553
 (a) Alpha=0.1, PD error=0.1554
 (a) Alpha=1, PD error=0.1554
 (a) Alpha=10, PD error=0.1554
 (a) Alpha=100, PD error=0.1554
 (a) Alpha=1000, PD error=0.1554
 (a) Alpha=10000, PD error=0.1554

===== Strategy 2 =====
```

Published with MATLAB® R2025b