ECE 271 Homework #3 - Programming Assignment

Arda Cankat Bati

Email: abati@ucsd.edu

Std. Id A53284500

All the MATLAB code and output (print statements, plots etc. are given at the end of this report.)

A)

In this assignment, we have two strategies. We choose a certain prior for the value of mu0, the prior we associate with the mean of the distribution. We also choose a prior diagonal covariance matrix for this prior mu. The diagonality means we consider the 64 features to be uncorrelated to each other. In the E0 covariance matrix, we assign a variance value to each individual feature as alpha*w_i. Weight values w_i are fixed, so the covariance matrix, and ultimately the formulas we get are all functions of alpha. When we assign low variance to our priors this means that we are more confident with our data. As the prior variance values increase, it means our mu0 priors vary a lot, so we are less certain about their accuracy. How much confidence we put in our priors is another way of saying how informative they are. If they are considered informative, then they should be affecting the outcome strongly. If they are considered very uninformative, they should effect the outcome as little as possible. This is just the case for the Bayesian solution we have used.

For the Bayesian Predictive solution we have the following formulas for the scalar case:

$$\mu_n = \underbrace{\frac{n\sigma_0^2}{\sigma^2 + n\sigma_0^2}}_{\alpha_n} \mu_{ML} + \underbrace{\frac{\sigma^2}{\sigma^2 + n\sigma_0^2}}_{1-\alpha_n} \mu_0$$

$$\frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}$$

The overall idea of these formulas also apply to the multivariate case. Here our prior of the mean is m0 and our covariance prior is sigma0. mn , which is the mean of the predictive distribution is a combination of the ML estimation (sample mean) and our prior. It can clearly be seen that as sigma0 increases, the solution favors the sample mean much more over our own priors.

For the dataset#1 and strategy 1, the error results of the predictive distribution can be seen on the first plot. We can see that for low alpha, the error probability is low, and as alpha increases error also increases. After a certain alpha value (around alpha = 1) the error probability becomes flat for later

alphas. This is because we assign such little weight to our priors that after a threshold the end result is always the same (the ML estimation of the mean dominates). For low alpha, our priors are more informative, so they contribute more to the overall predictive distribution. We see that for this part, when we rely less on our priors, we get higher error values. This due to our prior mu's being good estimations of the actual mu's of the test sets. The first DCT coefficient which is also called the DC component, is proportional to the average of all the input samples. It is also always positive. As Cheetah points are darker, and grass points are lighter, also considering the 0 is the darkest we can get, it is reasonable to assume mu=1 for FG and mu=3 for BG. To check this, in MATLAB I printed the average 1^{st} DCT coefficient value for each dataset and class. For the foreground datasets this value was around 1.3 and for background this value was around 2.9. Our own priors mu=1 for FG and mu=3 for BG are quite close to these values. Without ever seeing the datasets, we set sensible priors and therefore got better results when we rely on them. As we start to consider our priors less informative by increasing alpha, we our hindering our prediction therefore our error probability increases.

B)

The probability of error from ML procedure Is given in the print output of MATLAB (also on the plots.) For the dataset 1 and strategy 1, we see that ML procedure has higher error than the Bayesian predictive procedure for all alpha values. For the ML estimators we assume the underlying distribution of the data to be gaussian with a non random mu and covariance. We get these values from sample mean and sample covariance, and do not consider any of our priors in this process. As our priors in strategy 1 are very close to the actual dataset means, not taking them into account decreases the performance.

From the plot we can also see that the error rate of the Bayesian predictive procedure never converges that of the ML procedure. This is expected, the 1^{st} dataset shows this fact most clearly, as it has the least number of data points. For a lower number of data points and with good priors, predictive decision yields better results than ML procedure. Using ML estimators we represent the whole distribution with a single mu vector and sigma matrix. Therefore, in our decision function we losing some information from the dataset. However, for the Bayesian predictive procedure we aren't using singular non random values to represent the dataset. As we are considering the amalgamation of many models, we aren't losing information from the dataset.

C)

When we use the MAP Estimate, we are selecting a single prior model for mu0 and sigma0. In contrast, for part A we were choosing a weighted sum of different models of mu0 and sigma0. In MAP estimate, when summing over all the possible mu0 sigma0 models, we put a dirac delta function in the integral, there only use a single distribution. Going through the mathematics of the Bayes MAP decision rule, we get the following results from our course notes:

- ML-BDR
  - pick i if

$$i^*(x) = \arg\max_i P_{X|Y}\left(x \mid i; \theta_i^*\right) P_Y(i)$$
$$\text{where } \theta_i^* = \arg\max_\theta P_{X|Y}(D \mid i, \theta)$$

- Bayes MAP-BDR
  - pick i if

$$i^*(x) = \arg\max_i P_{X|Y}\left(x \mid i; \theta_i^{MAP}\right) P_Y(i)$$
$$\text{where } \theta_i^{MAP} = \arg\max_\theta P_{T|Y,\Theta}(D \mid i, \theta) P_{\Theta|Y}(\theta \mid i)$$

Here we can see that the MAP decision is very similar to ML decision rule. The only difference between them is that when deciding on the model parameters, we also take the prior distribution into account. This is because we view the model parameter as a random variable, instead of a fixed quantity. Here if we consider a prior θ distribution that is uniform, the prior for θ can be removed from the argmax function, as becomes constant. Then our solution becomes exactly the ML solution. Therefore, ML can be considered as a special case of MAP where our distribution of the prior is uniform. Uniform distribution has the highest variance of any possible distribution of random variables. So as the variance of our prior for θ increases, it becomes closer to a uniform distribution. For very high variance values, it practically converges to a uniform distribution. Overall, the difference between ML and MAP is evident only when we have a relatively small dataset

Increasing the variance of the θ prior corresponds to increasing the alpha values in our case. As alpha directly effects the covariance magnitude of our priors, ultimately it makes the prior distribution converge to uniform. By our previous argument, increasing alpha makes the priors less informative. Uniform is the most uninformative distribution we can propose for the underlying distribution. From this discussion it becomes clear that the MAP solution should converge to ML solution for high alpha values. We can see from the first plot, and all the other plots that the error rate of MAP converges to ML with increasing alpha. For low alpha, the error of MAP solution is less than ML. Although we consider only a single model, we still use the information from our prior mu values. As our prior estimated values for mu are better in strategy 1, using them always gives us an advantage. As the MAP solution converges ML, its error also increases and converges to ML error. For the first dataset, we also see that the Bayesian predictive solution has always lower error than MAP. As we consider a sum of different models in Bayesian prediction, we compensate for the possible offset our prior estimations have. However, in MAP, we have only one model, which increases the error when we don't have perfect prior estimations m0.

D)

The main differences between the datasets are the number of sample points. From dataset 1 to 4 the dataset size steadily increases. From the results we see that, overall, the ML error decreases with increasing size of datasets. The exception to this is the 2nd dataset, which has the smallest ML error. As

all the concepts we are dealing here are inherently random, results such as these can occur. Although dataset 2 has smaller size, the decision model we make out of it is more successful with our test data. These results could change for different training and test datasets. The important point here is pointing out the overall tendency here. The error rate tends to go down with increasing dataset size. As mentioned before, the error of MAP procedure always converges to ML, with increasing alpha. With increasing data size, we also observe that MAP and Bayesian Estimation results also become very similar. Although we consider multiple models in the Bayesian Est. Procedure, as we receive more data, the prior distribution becomes narrower. Although MAP estimation only uses a single model, as the dataset size increases, we become more and more confident with this model. Therefore, the results of MAP and Bayesian procedure become more similar to each other, which is reflected in their error rates.

E)

The only difference between strategy 1 and 2 is the prior value we assign to the 1st coefficient of the DCT. For strategy 1 we chose 1 and 3 for foreground and background respectively which was a reasonable choice. For strategy 2, we chose mu0 = 2, which is half the range of the 1st DCT coefficient, for both foreground and background cases. We can immediately see this is a worse choice than strategy one, as the darker cheetah and lighter grass will not have the same average values. As the 1st coefficient of the DCTs are proportional to the sample average of the image, we are not making a good choice. This is reflected in our error probability results. In contrast to the first strategy, the errors start with high probability for low alpha, and get lower with increasing alpha. We are actually benefiting from trusting our priors less by increasing their variance. As we consider our priors to be less informative, we get better results. While for the first data set the Bayesian solutions error is better than the MAP solution, for the later datasets they are mostly similar. This is again caused by the increase in dataset sizes. We also see that the ML estimation, which doesn't consider any priors at all, is generally better or equal to MAP and Bayesian Estimation as alpha changes. Even Bayesian estimation and MAP are meant to give better results than ML for low data point values, if we choose bad priors, they actually give worse results.

```
%**********************
% Arda Cankat Bati
% ECE 271A - Homework#2
%**********************

% BAYESIAN ESTIMATION

clear
clc
load('TrainingSamplesDCT_subsets_8.mat');
load('alpha.mat');

% ML Estimation Results from Part 2
ML_Error = zeros(1,4);

FGs = {D1_FG,D2_FG,D3_FG,D4_FG};
BGs = {D1_BG,D2_BG,D3_BG,D4_BG};

%*********** GETTING THE DATA AND ESTIMATING PRIORS **********

[cImageOld,colormap] = imread('cheetah.bmp');
cImage = im2double(cImageOld);
paddingType = 'replicate';
cImage = padarray(cImage,[4 4],paddingType,'pre');
cImage = padarray(cImage,[3 3],paddingType,'post');
% imshow(cImage); title('Original image with symmetric padding.');
% figure();

[cImageReal colormap] = imread('cheetah_mask.bmp');
cImageReal = double(cImageReal)/255;
Image_Size = size(cImageReal,2)*size(cImageReal,1);
FG_Sum = sum(sum(cImageReal));
BG_Sum = Image_Size - FG_Sum;

cImageOldX = size(cImageOld,1); cImageOldY = size(cImageOld,2);
cImageX = size(cImage,1); cImageY = size(cImage,2);
A = [0  1  5  6  14  15  27  28 2  4  7  13  16  26  29  42 3  8  12
 17  25  30  41  43 9  11  18  24  31  40  44  53 10  19  23  32  39
 45  52  54 20  22  33  38  46  51  55  60 21  34  37  47  50  56  59
 61 35  36  48  49  57  58  62  63];
A = A + 1;
decisionImage64 = zeros(size(cImageOld,1),size(cImageOld,2));
points = zeros(cImageOldX*cImageOldY,64);
point = zeros(64,1);
count = 1;
for i = 1:cImageOldX
    for j = 1:cImageOldY
        temp = (dct2(cImage(i:i+7, j:j+7)))';
        vectorDct= temp(:);
        point(A) = vectorDct;
        points(count,:) = point';
        count = count + 1;
```

```
    end
end

load('Prior_1.mat');
str1_mu0_FG = mu0_FG;
str1_mu0_BG = mu0_BG;
str1_w0 = W0;

clear mu0_FG
clear mu0_BG
clear W0

load('Prior_2.mat');
str2_mu0_FG = mu0_FG;
str2_mu0_BG = mu0_BG;

str2_w0 = W0;

mu0_FG = zeros(2,64);
mu0_BG = zeros(2,64);

mu0_FG(1,:) = str1_mu0_FG;
mu0_FG(2,:) = str2_mu0_FG;
mu0_BG(1,:) = str1_mu0_BG;
mu0_BG(2,:) = str2_mu0_BG;

%************ LOOP FOR PART 1 ********************

for loop = 1:4

    FG = cell2mat(FGs(loop));
    BG = cell2mat(BGs(loop));
    FG_size = size(FG,1);
    BG_size = size(BG,1);
    sampleSize = FG_size + BG_size;

    CPrior = FG_size/(sampleSize);
    NCPrior = BG_size/(sampleSize);

    mean1st_FG = mean(FG(:,1));
    mean1st_BG = mean(BG(:,1));

    fprintf('For dataset#%d mean of the 1st foreround DCT coefficient
 is: %f\n',loop,mean1st_FG);
    fprintf('For dataset#%d mean of the 1st background DCT coefficient
 is: %f\n',loop,mean1st_BG);

    %******** SAMPLE MEAN & VARIANCE CALCULATION***********

    % Sample Mean calculation
    sampleMeanFG = mean(FG,1)';
    sampleMeanBG = mean(BG,1)';
```

```
   %Covariance matrix calculation for 64 dimensional prob.
distribution
   CovMtxFG = cov(FG);
   CovMtxBG = cov(BG);

   cov0 = zeros(2,9,64,64);

   for i=1:9
       cov0(1,i,:,:) = diag(alpha(i)*str1_w0);
       cov0(2,i,:,:) = diag(alpha(i)*str2_w0);
   end

   for strategy = 1:2

       for alpha_count = 1:9

           %fprintf('Dataset %d, Strategy %d,
Alpha(%d)\n',loop,strategy,alpha_count);

           n = FG_size;
           E0 = squeeze(cov0(strategy,alpha_count,:,:));
           E = CovMtxFG;
           mn_hat = (sampleMeanFG);
           m0 = (mu0_FG(strategy,:))';

           m1_FG = (E0/(E0 + E/n))*mn_hat + ((E/n)/(E0 + E/n))*m0;
           E1_FG = (E0/(E0 + E/n))*(E/n);

           n = BG_size;
           E0 = squeeze(cov0(strategy,alpha_count,:,:));
           E = CovMtxBG;
           mn_hat = (sampleMeanBG);
           m0 = (mu0_BG(strategy,:))';

           m1_BG = (E0/(E0 + E/n))*mn_hat + ((E/n)/(E0 + E/n))*m0;
           E1_BG = (E0/(E0 + E/n))*(E/n);

           E_FG = CovMtxFG + E1_FG;
           E_BG = CovMtxBG + E1_BG;


           %******* CLASSIFYING EACH PIXEL IN THE TEST IMAGE ********

           decisionImage64 =
zeros(size(cImageOld,1),size(cImageOld,2));

           %All 64 Features
           point = zeros(64,1);
           invFG = inv(E_FG);
           detFG = det(E_FG);
           invBG = inv(E_BG);
           detBG = det(E_BG);

           count = 1;
```

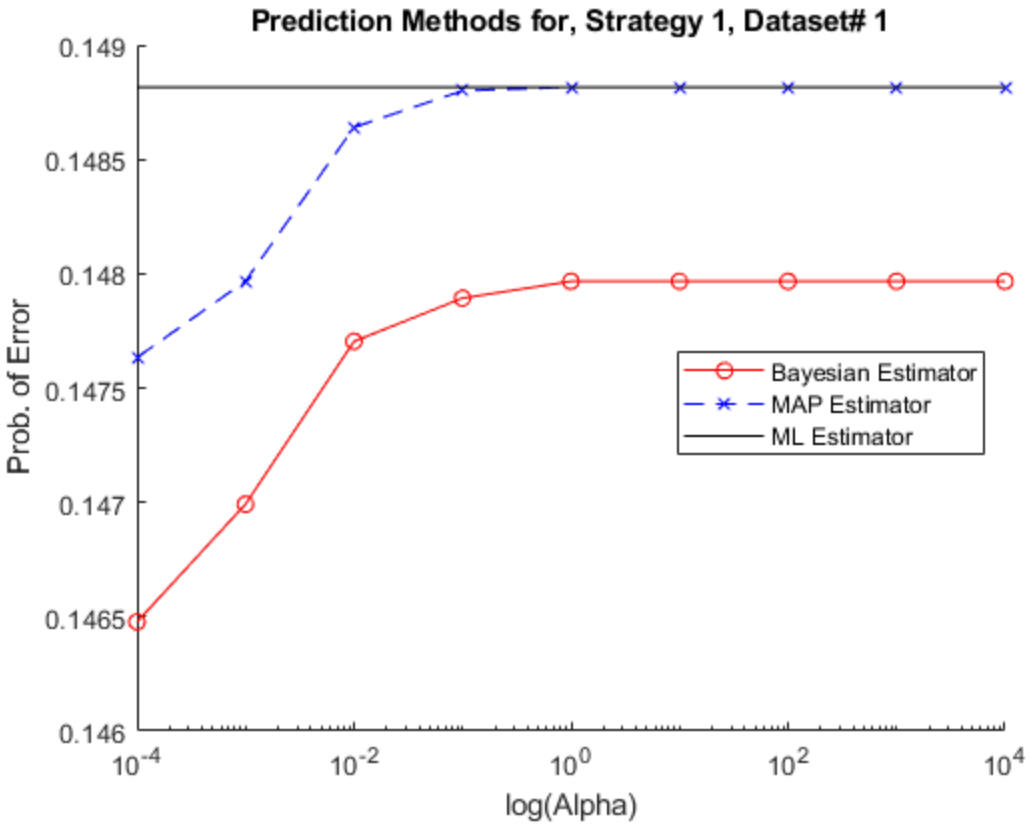
```
            for i = 1:cImageOldX
                for j = 1:cImageOldY
                    point = points(count,:)';
                    mhbDistFG = ((point - m1_FG)')*(invFG)*(point -
 m1_FG);
                    alphaFG = log(((2*pi)^64)*detFG) - 2*log(CPrior);
                    mhbDistBG = ((point - m1_BG)')*(invBG)*(point -
 m1_BG);
                    alphaBG = log(((2*pi)^64)*detBG) - 2*log(NCPrior);
                    [M,decision] = min([(mhbDistBG + alphaBG)
 (mhbDistFG + alphaFG)]);
                    decision = decision - 1;
                    decisionImage64(i,j) = decision;
                    count = count + 1;
                end
            end

            %*************************************************

            errorMaskBG = decisionImage64 - cImageReal;
            %FG misclassified as BG / total true FG
            beta64 = sum(sum(errorMaskBG == -1)) / FG_Sum; %False
 Negative --> beta
            %BG misclassified as FG / total true BG
            alpha64 = sum(sum(errorMaskBG == 1)) / BG_Sum;  %False
 Positive --> alpha
            PE_Bayes(strategy,loop,alpha_count) = CPrior * beta64 +
 NCPrior * alpha64;

    %         I = mat2gray(decisionImage64,[0 1]);
    %         imshow(I); title(sprintf('Prediction image with alpha=
 %f',alpha));
    %         figure()

        end
    end
end

% ********** LOOP FOR PART 2 ******************

print('');

for loop=1:4

    FG = cell2mat(FGs(loop));
    BG = cell2mat(BGs(loop));
    FG_size = size(FG,1);
    BG_size = size(BG,1);
    sampleSize = FG_size + BG_size;

    CPrior = FG_size/(sampleSize);
    NCPrior = BG_size/(sampleSize);

    %******** SAMPLE MEAN & VARIANCE CALCULATION**********
```

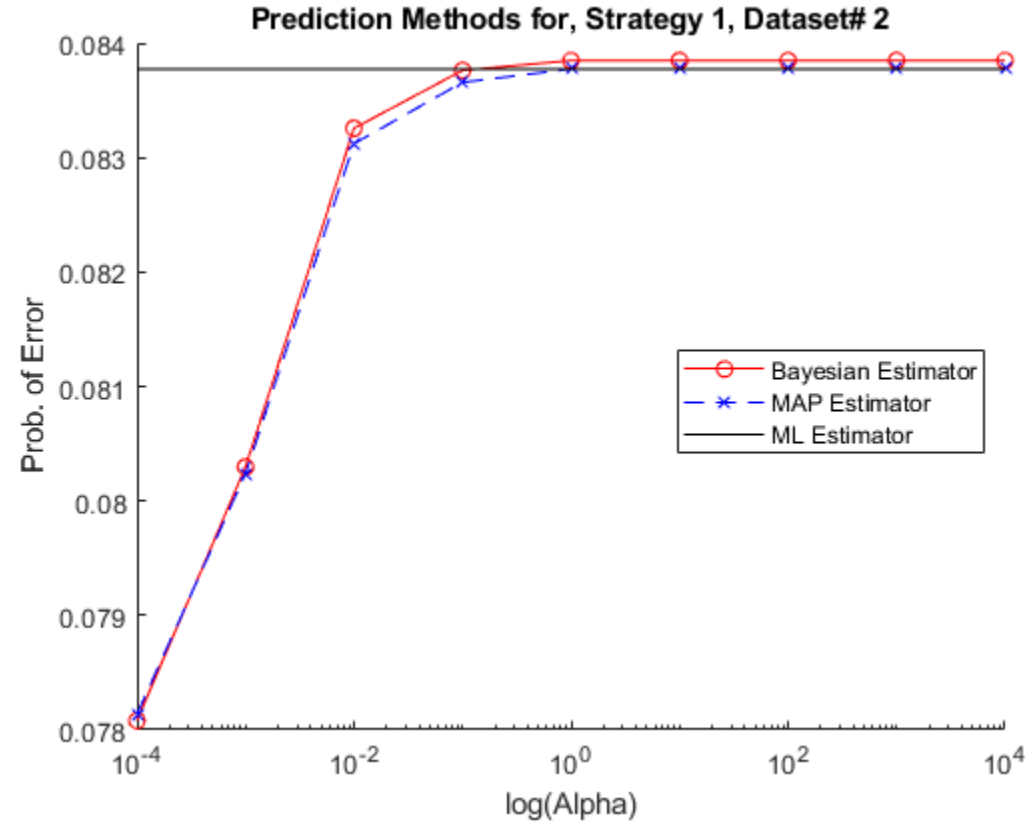
```
    % Sample Mean calculation
    sampleMeanFG = mean(FG,1)';
    sampleMeanBG = mean(BG,1)';

    %Covariance matrix calculation for 64 dimensional prob.
 distribution
    CovMtxFG = cov(FG);
    CovMtxBG = cov(BG);

    %All 64 Features
    point = zeros(64,1);
    count = 1;
    for i = 1:cImageOldX
        for j = 1:cImageOldY
            point = points(count,:)';
            mhbDistFG = ((point -
 sampleMeanFG)')*(inv(CovMtxFG))*(point - sampleMeanFG);
            alphaFG = log(((2*pi)^64)*det(CovMtxFG)) - 2*log(CPrior);
            mhbDistBG = ((point -
 sampleMeanBG)')*(inv(CovMtxBG))*(point - sampleMeanBG);
            alphaBG = log(((2*pi)^64)*det(CovMtxBG)) - 2*log(NCPrior);
            [M,decision] = min([(mhbDistBG + alphaBG) (mhbDistFG +
 alphaFG)]);
            decision = decision - 1;
            decisionImage64(i,j) = decision;
            count = count + 1;
        end
    end

    %******* PRINTING THE FINAL BLACK & WHITE IMAGES   *******

%     I = mat2gray(decisionImage64,[0 1]);
%     imshow(I); title('64 Features prediction with W=Cheetah,
 B=NoCheetah');
%     figure()

    %******* TOTAL ERROR CALCULATION FOR THE TWO CASES *********

    % 64 Gaussian Features Error
    errorMaskBG = decisionImage64 - cImageReal;
    %FG misclassified as BG / total true FG
    beta64 = sum(sum(errorMaskBG == -1)) / FG_Sum; %False Negative --
> beta
    %BG misclassified as FG / total true BG
    alpha64 = sum(sum(errorMaskBG == 1)) / BG_Sum;  %False Positive --
> alpha
    ProbOfError64 = CPrior * beta64 + NCPrior * alpha64;
    fprintf('ML estimation error probability for dataset#%d is: %f
\n',loop,ProbOfError64);
    ML_Error(loop) = ProbOfError64;

end
```

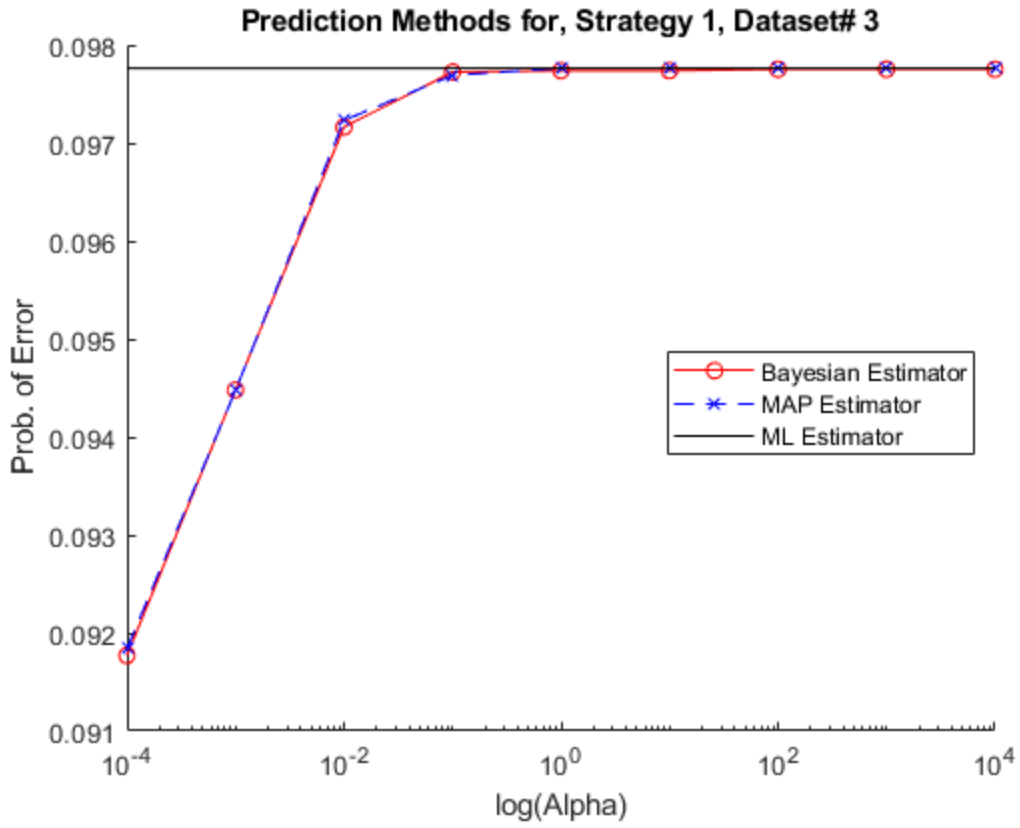
```
% ************ LOOP FOR PART 3 ********************

for strategy = 1:2

    for loop = 1:4

        FG = cell2mat(FGs(loop));
        BG = cell2mat(BGs(loop));
        FG_size = size(FG,1);
        BG_size = size(BG,1);
        sampleSize = FG_size + BG_size;

        CPrior = FG_size/(sampleSize);
        NCPrior = BG_size/(sampleSize);


        %******** SAMPLE MEAN & VARIANCE CALCULATION***********

        % Sample Mean calculation
        sampleMeanFG = mean(FG,1)';
        sampleMeanBG = mean(BG,1)';

        %Covariance matrix calculation for 64 dimensional prob.
 distribution
        CovMtxFG = cov(FG);
        CovMtxBG = cov(BG);

        cov0 = zeros(2,9,64,64);

        for i=1:9
            cov0(1,i,:,:) = diag(alpha(i)*str1_w0);
            cov0(2,i,:,:) = diag(alpha(i)*str2_w0);
        end

        for alpha_count = 1:9

            %fprintf('Dataset %d, Strategy %d,
 Alpha(%d)\n',loop,strategy,alpha_count);

            n = FG_size;
            E0 = squeeze(cov0(strategy,alpha_count,:,:));
            E = CovMtxFG;
            mn_hat = (sampleMeanFG);
            m0 = (mu0_FG(strategy,:))';

            m1_FG = (E0/(E0 + E/n))*mn_hat + ((E/n)/(E0 + E/n))*m0;

            n = BG_size;
            E0 = squeeze(cov0(strategy,alpha_count,:,:));
            E = CovMtxBG;
            mn_hat = (sampleMeanBG);
            m0 = (mu0_BG(strategy,:))';
```

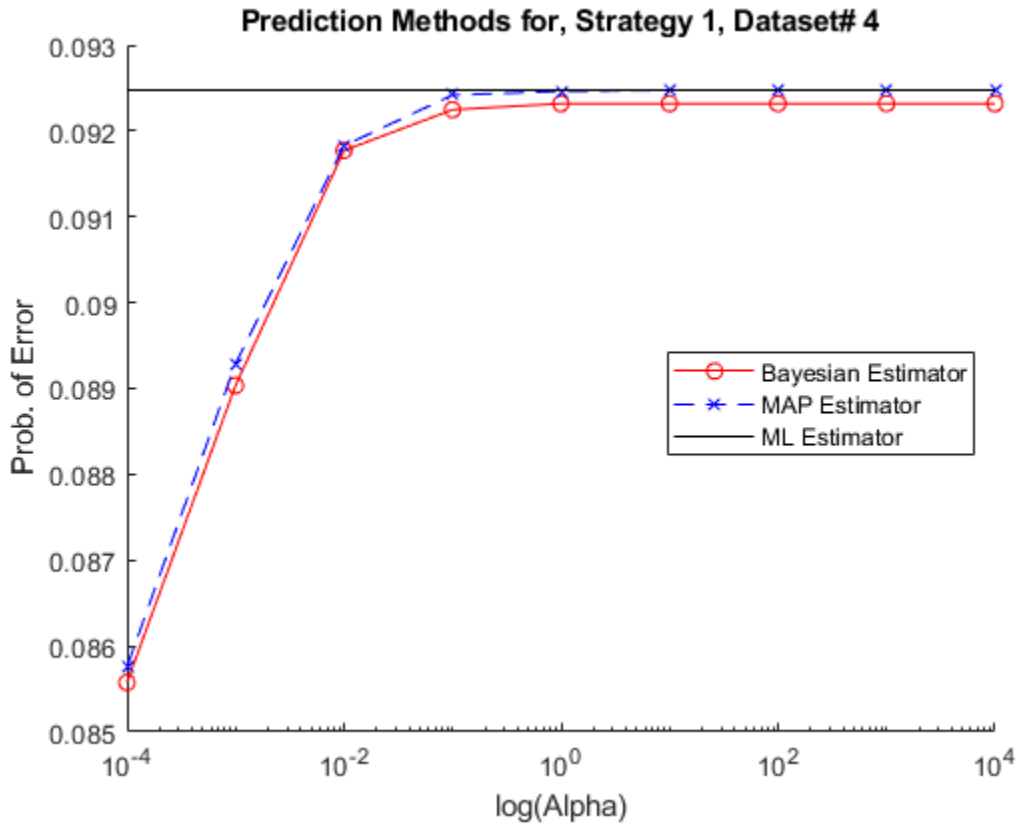
```
            m1_BG = (E0/(E0 + E/n))*mn_hat + ((E/n)/(E0 + E/n))*m0;

            E_FG = CovMtxFG;
            E_BG = CovMtxBG;

            %******* CLASSIFYING EACH PIXEL IN THE TEST IMAGE ********

            decisionImage64 =
 zeros(size(cImageOld,1),size(cImageOld,2));

            %All 64 Features
            point = zeros(64,1);
            invFG = inv(E_FG);
            detFG = det(E_FG);
            invBG = inv(E_BG);
            detBG = det(E_BG);

            count = 1;
            for i = 1:cImageOldX
                for j = 1:cImageOldY
                    point = points(count,:)';
                    mhbDistFG = ((point - m1_FG)')*(invFG)*(point -
 m1_FG);
                    alphaFG = log(((2*pi)^64)*detFG) - 2*log(CPrior);
                    mhbDistBG = ((point - m1_BG)')*(invBG)*(point -
 m1_BG);
                    alphaBG = log(((2*pi)^64)*detBG) - 2*log(NCPrior);
                    [M,decision] = min([(mhbDistBG + alphaBG)
 (mhbDistFG + alphaFG)]);
                    decision = decision - 1;
                    decisionImage64(i,j) = decision;
                    count = count + 1;
                end
            end

            errorMaskBG = decisionImage64 - cImageReal;
            %FG misclassified as BG / total true FG
            beta64 = sum(sum(errorMaskBG == -1)) / FG_Sum; %False
 Negative --> beta
            %BG misclassified as FG / total true BG
            alpha64 = sum(sum(errorMaskBG == 1)) / BG_Sum;  %False
 Positive --> alpha
            PE_MAP(strategy,loop,alpha_count) = CPrior * beta64 +
 NCPrior * alpha64;

%           I = mat2gray(decisionImage64,[0 1]);
%           imshow(I); title(sprintf('Prediction image with alpha=
 %f',alpha));
%           figure()
        end
    end
end
```

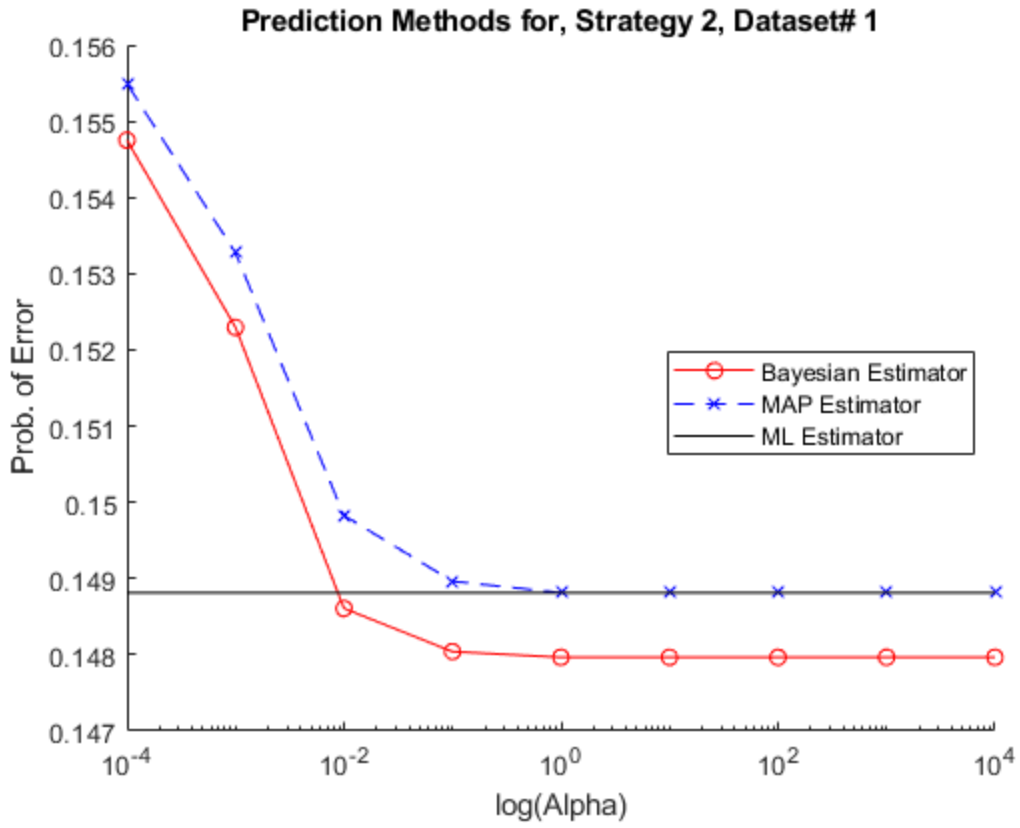
```
% ********* PLOTTING THE RESULTS **************

for i = 1:2
    for j = 1:4
        y1 = squeeze(PE_Bayes(i,j,:))';
        y2 = squeeze(PE_MAP(i,j,:))';
        y3(1:9) = ML_Error(j);
        hold on
        p1 = plot(alpha,y1,'ro-'); L1 = "Bayesian Estimator";
        p2 = plot(alpha,y2,'bx--'); L2 = "MAP Estimator";
        p3 = plot(alpha,y3,'k');      L3 = "ML Estimator";
        lgd = legend([p1,p2, p3], [L1, L2, L3]);
        lgd.Position = [0.75 0.5 0 0];
        hold off
        set(gca,'XScale', 'log')
        title(sprintf('Prediction Methods for, Strategy %d, Dataset#
 %d',i,j));
        xlabel('log(Alpha)')
        ylabel('Prob. of Error')
        figure()
    end
end

% ***** PRINT OUTPUT FROM THE CODE IS BELOW ***************

For dataset#1 mean of the 1st foreround DCT coefficient is: 1.329216
For dataset#1 mean of the 1st background DCT coefficient is: 2.931621
For dataset#2 mean of the 1st foreround DCT coefficient is: 1.288431
For dataset#2 mean of the 1st background DCT coefficient is: 2.923327
For dataset#3 mean of the 1st foreround DCT coefficient is: 1.231218
For dataset#3 mean of the 1st background DCT coefficient is: 2.925081
For dataset#4 mean of the 1st foreround DCT coefficient is: 1.264229
For dataset#4 mean of the 1st background DCT coefficient is: 2.920069
ML estimation error probability for dataset#1 is: 0.148815
ML estimation error probability for dataset#2 is: 0.083773
ML estimation error probability for dataset#3 is: 0.097763
ML estimation error probability for dataset#4 is: 0.092476
```

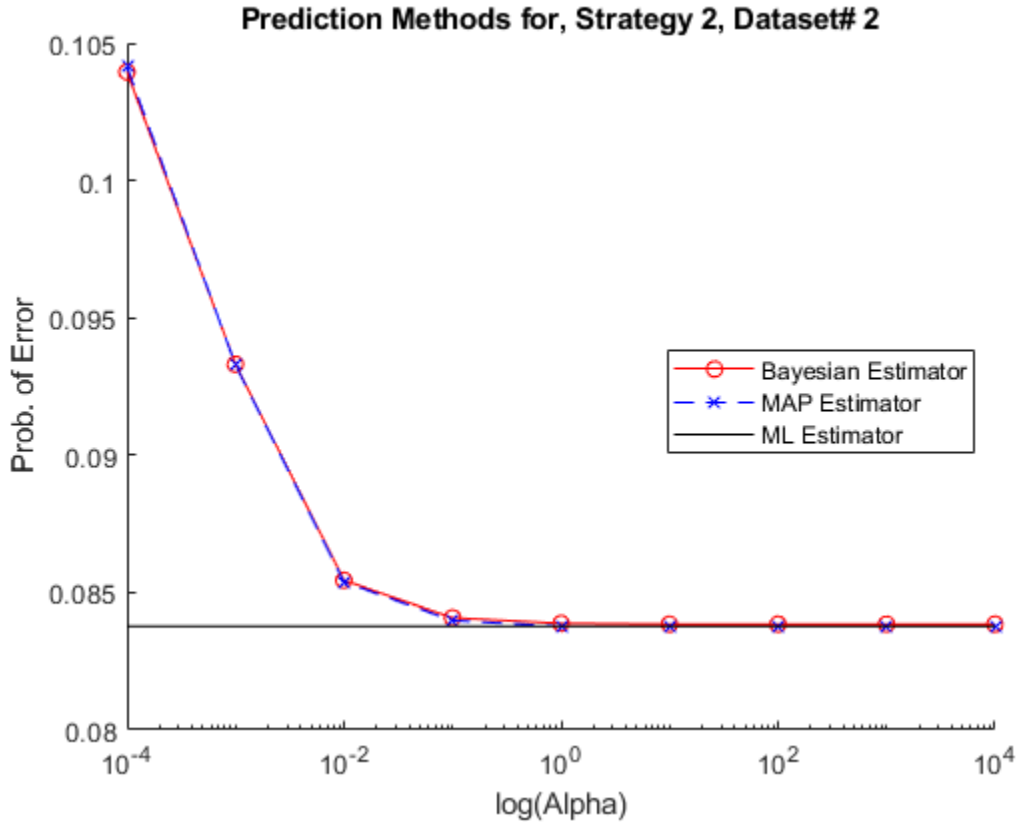

Prediction Methods for, Strategy 1, Dataset# 1
0.149
0.1485
0.148
0.1475
0.147
0.1465
0.146
Prob. of Error
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator

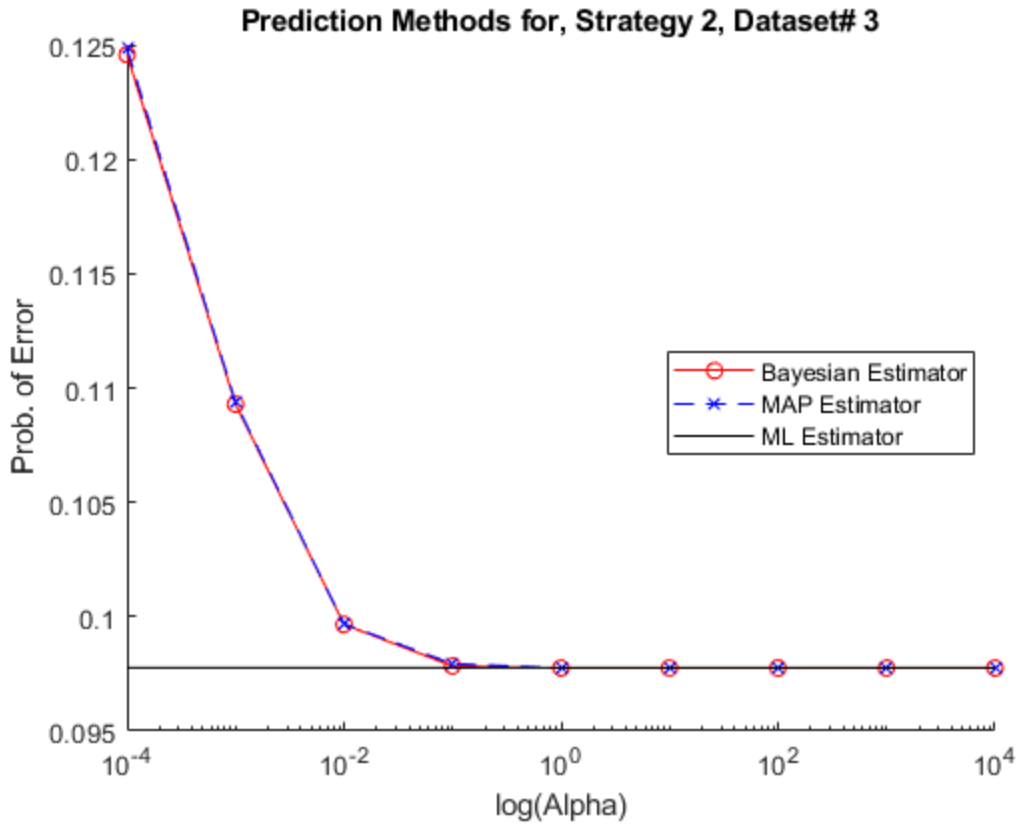

Prediction Methods for, Strategy 1, Dataset# 2
0.084
0.083
0.082
0.081
0.08
0.079
0.078
Prob. of Error
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator

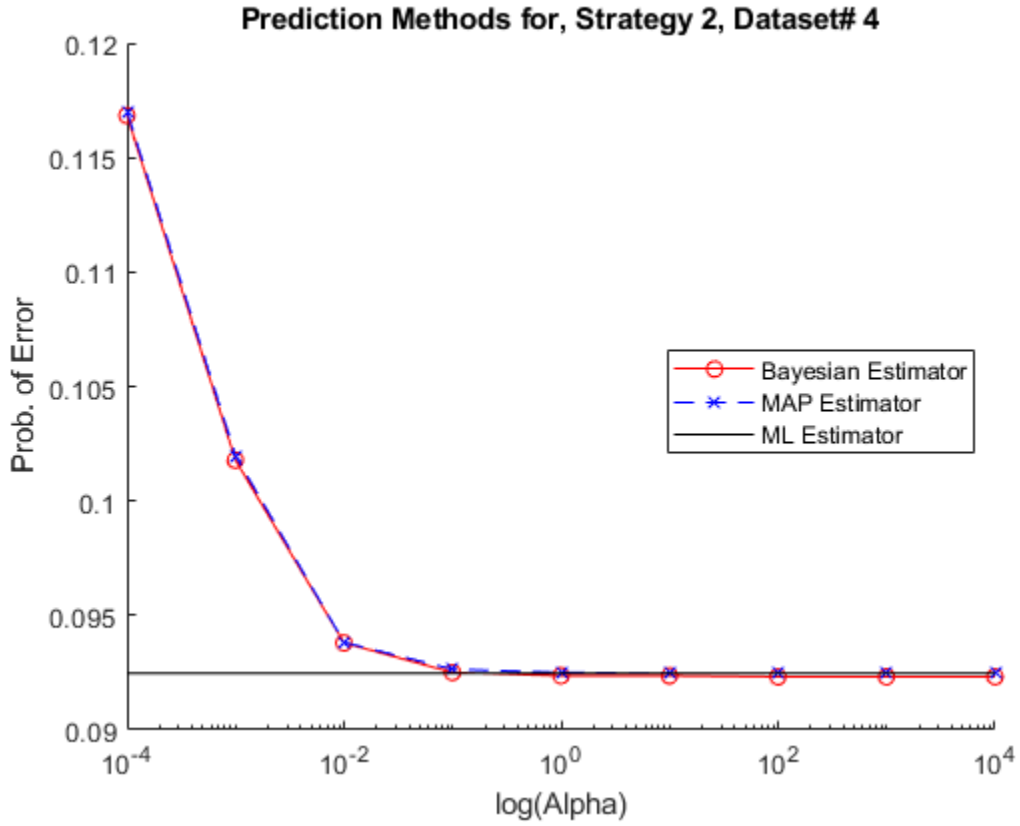

Prediction Methods for, Strategy 1, Dataset# 3
0.098
0.097
0.096
0.095
0.094
0.093
0.092
0.091
Prob. of Error
Bayesian Estimator
MAP Estimator
ML Estimator
10^-4
10^-2
10^0
10^2
10^4
log(Alpha)


Prediction Methods for, Strategy 1, Dataset# 4
0.093
0.092
0.091
0.09
0.089
0.088
0.087
0.086
0.085
Prob. of Error
Bayesian Estimator
MAP Estimator
ML Estimator
10^-4
10^-2
10^0
10^2
10^4
log(Alpha)


Prediction Methods for, Strategy 2, Dataset# 1
Prob. of Error
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator
0.156
0.155
0.154
0.153
0.152
0.151
0.15
0.149
0.148
0.147
10^-4
10^-2
10^0
10^2
10^4


Prediction Methods for, Strategy 2, Dataset# 2
Prob. of Error
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator
0.105
0.1
0.095
0.09
0.085
0.08
10^-4
10^-2
10^0
10^2
10^4


Prediction Methods for, Strategy 2, Dataset# 3
Prob. of Error
0.125
0.12
0.115
0.11
0.105
0.1
0.095
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator


Prediction Methods for, Strategy 2, Dataset# 4
Prob. of Error
0.12
0.115
0.11
0.105
0.1
0.095
0.09
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
log(Alpha)
Bayesian Estimator
MAP Estimator
ML Estimator

Published with MATLAB® R2018b