Name: Neil Chen
PID: A59013393

December 1, 2021

ECE 271A Homework 3

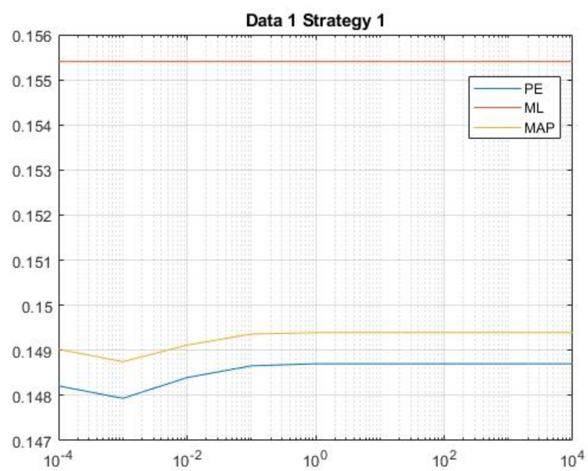
1. This week we will continue trying to classify our cheetah example. Once again we use the decomposition into 8x8 image blocks, compute the DCT of each block, and zig-zag scan. We also continue to assume that the class-conditional densities are multivariate Gaussians of 64 dimensions. The goal is to understand the benefits of a Bayesian solution.

   (a) Consider the training set $\mathbf{D_1}$ and **strategy 1**. For each class, compute the covariance $\sum$ of the classconditional, and the posterior μ_1, and $\sum_1$ of $P_{\mu|T}(\mu|D_1) = G(\mu, \mu_1, \sum_1)$. 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 α. 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 α. Can you explain the results?

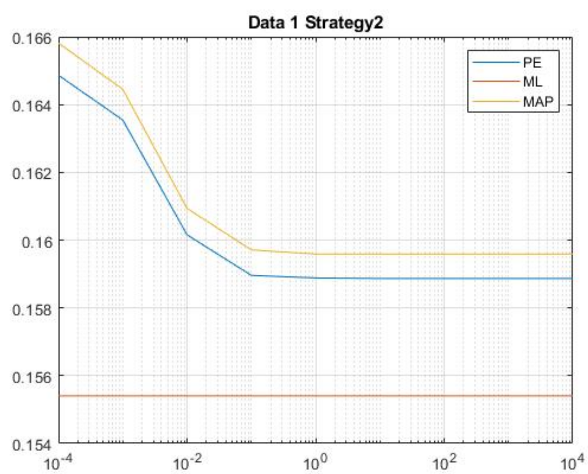
Solution.
For **Predictive Distribution method**, we get our training data sets from the prior predictive distribution, i.e., a collection of data sets generated from the model. After we have obtained the posterior distributions of the parameters, we can now use the posterior distributions to generate future data from the model. From the formula, we can see that the posterior distribution is a nothing more than a Gaussian distribution of μs, with μ_n and σ_n^2 unknown. However, we can obtain μ_n and σ_n^2 from the formulas mentioned in the lectures. It is worth noting that there is a crucial variable in the formula, i.e., σ_0^2. σ_0^2 is dependent upon our choice of α. If we change the value of α, then the value of σ_0^2 will change accordingly. However, when the value of α is greater than 1, the results only change subtly. α 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 α 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.

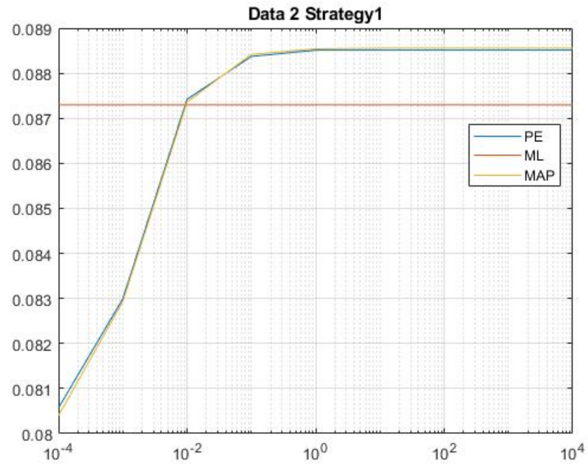


(a)

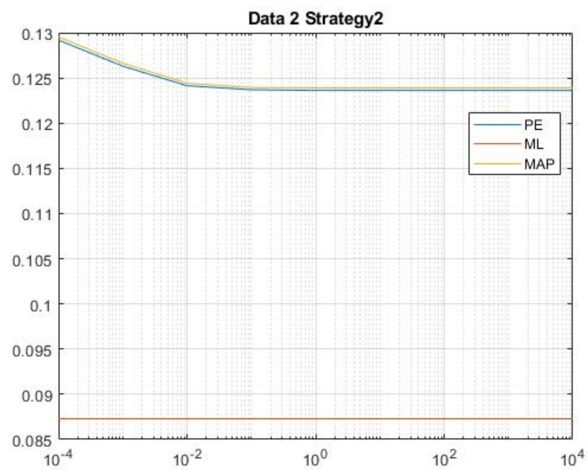


(b)

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

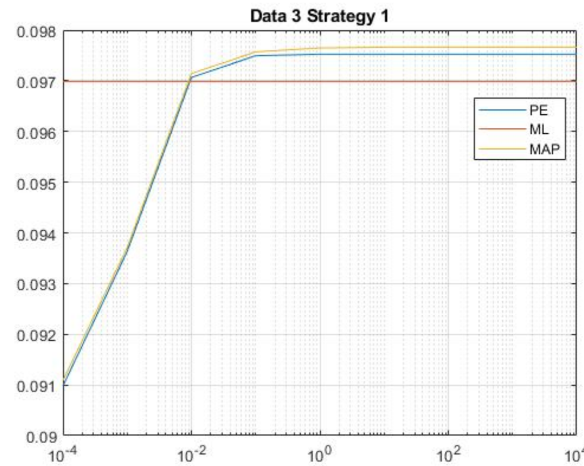


(a)

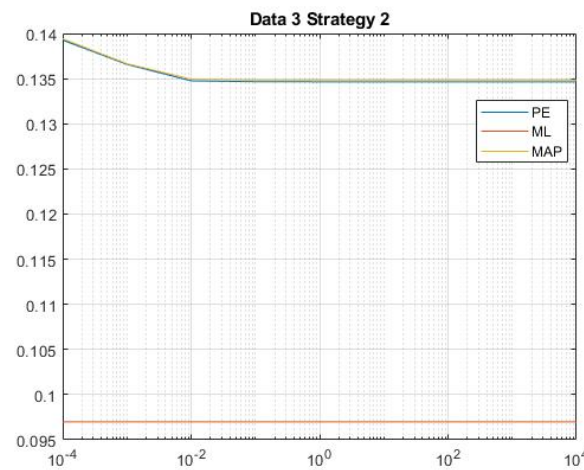


(b)

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

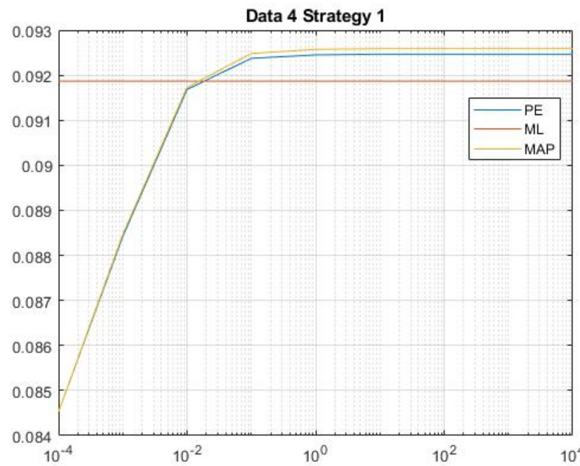


(a)

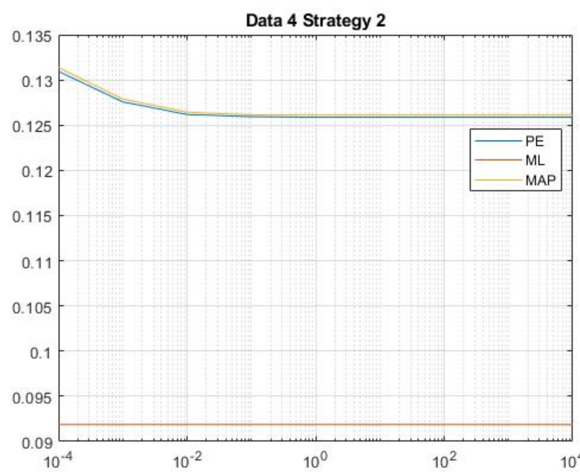


(b)

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 μ, 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 μ as a constant variable instead of a function of μ, hence there is only one μ, i.e., μ_{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, $i = 2, 3, ..., 4$. 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*, μ_0 is smaller for cheetah class ($\mu_0 = 1$) and larger for grass class ($\mu_0 = 3$), whereas for *Strategy 2*, μ_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

```
clear
clc

% ---- PE METHOD ----

% load the data
load("Alpha.mat");  % 1 x 9 vector
load("Prior_2.mat");    % Strategy 1 (W0 mu0_FG mu0_BG) %TODO: change
load("TrainingSamplesDCT_subsets_8.mat", "D4_BG", "D4_FG"); % D1: cheetah & grass
load("ZigZagVec.mat")   % 1 x 64 vector
load("cheetahMat.txt")  % our image
load("cheetah_mask.mat")    % our ideal mask

% get the size of the image
[m, n] = size(cheetahMat);

% create a matrix
m = m - 7;
n = n - 7;
totalPixels = m * n;

% # of samples
[n_cheetah, ~] = size(D4_FG);
[n_grass, ~]   = size(D4_BG);
totalSamples = (n_cheetah + n_grass);

% calculate the priors
Prior_cheetah = n_cheetah / totalSamples;
Prior_grass   = n_grass   / totalSamples;

% calculate the mu_MLs
mu_ML_cheetah = mean(D4_FG);
mu_ML_grass   = mean(D4_BG);

% calculate the covs
```

```
cov_cheetah = cov(D4_FG);
cov_grass   = cov(D4_BG);

% results
errorPE_D4P2 = zeros(1, 9);  %TODO: change

% Prior 1
for d = 1 : 9
    % calculate the sigma_0
    sigma_0 = diag(alpha(d) .* W0);

    % calculate the mu_ns
    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;
    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;

    % calculate the sigma_ns
    sigma_n_cheetah = 1 ./ ((1 ./ cov_cheetah) + (n_cheetah ./ cov_cheetah));
    sigma_n_grass   = 1 ./ ((1 ./ cov_grass)   + (n_grass   ./ cov_grass));

    % calculate SIGMAs (sigma + sigma_n) and their inverses
    SIGMA_cheetah = sigma_n_cheetah + cov_cheetah;
    SIGMA_grass   = sigma_n_grass   + cov_grass;
    SIGMAInv_cheetah = inv(SIGMA_cheetah);
    SIGMAInv_grass   = inv(SIGMA_grass);

    % calculate the coefficients
    deno_cheetah = (sqrt(((2 * pi) ^ 64) * det(SIGMA_cheetah)));
    deno_grass   = (sqrt(((2 * pi) ^ 64) * det(SIGMA_grass)));

    % calculate DCT2 (64D)
    error = 0;  % reset the error
    for i = 1 : m
```

```
        parfor j = 1 : n
            Block = cheetahMat(i : i + 7, j : j + 7);
            Block_DCT = dct2(Block, 8, 8);
            V = Block_DCT(:).';
            X = zeros(1, 64);
            % mapping
            for k = 1 : 64
                X(ZigZagVec(k)) = V(k);
            end

            % calculate the i*(x)
            P_X_likelihood_cheetah = exp(-0.5 .* (X - mu_n_cheetah) * (SIGMAInv_cheetah) * (X - mu_n_cheetah).') ./ deno_cheetah;
            P_X_likelihood_grass   = exp(-0.5 .* (X - mu_n_grass) * (SIGMAInv_grass)     * (X - mu_n_grass).')    ./ deno_grass;
            P_X_cheetah = P_X_likelihood_cheetah * Prior_cheetah;
            P_X_grass   = P_X_likelihood_grass   * Prior_grass;

            % based on the results, decide whether the pixel is grass or cheetah
            if (P_X_grass > P_X_cheetah)
                result = 0;
            else
                result = 1;
            end

            % compare my result with the ideal mask
            if (result ~= cheetah_mask(i, j))
                error = error + 1;
            end
        end
    end
    errorPE_D4P2(1, d) = error / totalPixels;
end
```

Code for ML

```
clear
clc

% --- ML METHOD ---

% load the data
load("Alpha.mat");  % 1 x 9 vector
load("TrainingSamplesDCT_subsets_8.mat", "D4_BG", "D4_FG"); % D1: cheetah & grass
load("ZigZagVec.mat")   % 1 x 64 vector
load("cheetahMat.txt")  % our image
load("cheetah_mask.mat")     % our ideal mask

% get the size of the image
[m, n] = size(cheetahMat);
m = m - 7;
n = n - 7;
totalPixels = m * n;
maskRes = zeros(m, n);

% # of samples
[n_cheetah, ~] = size(D4_FG);
[n_grass, ~]   = size(D4_BG);
totalSamples = (n_cheetah + n_grass);

% calculate the priors
Prior_cheetah = n_cheetah / totalSamples;
Prior_grass   = n_grass   / totalSamples;

% mus and covs
mu_cheetah = mean(D4_FG);
mu_grass   = mean(D4_BG);
cov_cheetah = cov(D4_FG);
cov_grass   = cov(D4_BG);

% calculate SIGMAs (sigma) and their inverses
SIGMA_cheetah = cov_cheetah;
```

```
SIGMA_grass     = cov_grass;
SIGMAInv_cheetah = inv(SIGMA_cheetah);
SIGMAInv_grass   = inv(SIGMA_grass);

% calculate the coefficients
deno_cheetah = (sqrt(((2 * pi)^64) * det(SIGMA_cheetah)));
deno_grass   = (sqrt(((2 * pi)^64) * det(SIGMA_grass)));

% results
errorML_D4 = zeros(1, 9);  %TODO: change

for d = 1 : 1

    % calculate DCT2 (64D)
    error = 0;
    for i = 1 : m
        parfor j = 1 : n
            Block = cheetahMat(i : i + 7, j : j + 7);
            Block_DCT = dct2(Block, 8, 8);
            V = Block_DCT(:).';
            X = zeros(1, 64);

            % mapping
            for k = 1 : 64
                X(ZigZagVec(k)) = V(k);
            end

            % calculate the probabilities
            % G(X, mu_n, SIGMA)
            P_X_likelihood_cheetah = exp(-0.5 * (X -
                mu_cheetah) * (SIGMAInv_cheetah) * (X -
                mu_cheetah).') / deno_cheetah;
            P_X_likelihood_grass   = exp(-0.5 * (X - mu_grass)
                 * (SIGMAInv_grass) * (X - mu_grass).') /
                deno_grass;
            P_X_cheetah = P_X_likelihood_cheetah *
                Prior_cheetah;
            P_X_grass   = P_X_likelihood_grass   * Prior_grass
                ;
```

```

            % calculate the error
            if (P_X_grass > P_X_cheetah)
                result = 0;
            else
                result = 1;
            end

            % compare my result with the ideal mask
            if (result ~= cheetah_mask(i, j))
                error = error + 1;
            end
        end
    end
    errorML_D4(1, d) = error / totalPixels;
end

for k = 2 : 9
    errorML_D4(1, k) = errorML_D4(1, 1);
end

save('errorML_D4.mat', 'errorML_D4')
```

Code for MAP

```
clear
clc

% —— MAP METHOD ——

% load the data
load("Alpha.mat"); % 1 x 9 vector
load("Prior_2.mat"); % Strategy 1 (W0 mu0_FG mu0_BG)
load("TrainingSamplesDCT_subsets_8.mat", "D1_BG", "D1_FG"); % D1: cheetah & grass
load("ZigZagVec.mat") % 1 x 64 vector
load("cheetahMat.txt") % our image
load("cheetah_mask.mat") % our ideal mask

```

```
% get the size of the image
[m, n] = size(cheetahMat);
m = m - 7;
n = n - 7;
maskRes = zeros(m, n);
totalPixels = m * n;

% # of samples
[n_cheetah, ~] = size(D1_FG);
[n_grass, ~]   = size(D1_BG);
totalSamples = (n_cheetah + n_grass);

% calculate the priors
Prior_cheetah = n_cheetah / totalSamples;
Prior_grass   = n_grass   / totalSamples;

% mu_MLs and covs
mu_ML_cheetah = mean(D1_FG);
mu_ML_grass   = mean(D1_BG);
cov_cheetah = cov(D1_FG);
cov_grass   = cov(D1_BG);

% calculate the sigma_0
d = 6;  % which alpha am I using
sigma_0 = diag(alpha(d) .* W0);

% calculate the mu_ns
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;
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;

% calculate SIGMAs (sigma) and their inverses
SIGMA_cheetah = cov_cheetah;
SIGMA_grass   = cov_grass;
SIGMAInv_cheetah = inv(SIGMA_cheetah);
SIGMAInv_grass   = inv(SIGMA_grass);
```

```
% calculate the coefficients
deno_cheetah = (sqrt(((2 * pi)^64) * det(SIGMA_cheetah)));
deno_grass   = (sqrt(((2 * pi)^64) * det(SIGMA_grass)));

% calculate DCT2 (64D)
error = 0;
for i = 1 : m
    for j = 1 : n
        Block = cheetahMat(i : i + 7, j : j + 7);
        Block_DCT = dct2(Block, 8, 8);
        V = Block_DCT(:).';
        X = zeros(1, 64);

        % mapping
        for k = 1 : 64
            X(ZigZagVec(k)) = V(k);
        end

        % calculate the probabilities
        % G(X, mu_n, SIGMA)
        P_X_likelihood_cheetah = exp(-0.5 * (X - mu_n_cheetah) * (SIGMAInv_cheetah) * (X - mu_n_cheetah).') / deno_cheetah;
        P_X_likelihood_grass   = exp(-0.5 * (X - mu_n_grass) * (SIGMAInv_grass) * (X - mu_n_grass).') / deno_grass;
        P_X_cheetah = P_X_likelihood_cheetah * Prior_cheetah;
        P_X_grass   = P_X_likelihood_grass   * Prior_grass;

        % based on the results, decide whether the pixel is grass or cheetah
        if (P_X_grass > P_X_cheetah)
            maskRes(i, j) = 0;
        else
            maskRes(i, j) = 1;
        end

        % calculate the error
```

```
            if (P_X_grass > P_X_cheetah)
                result = 0;
            else
                result = 1;
            end

            % calculate the error rate
            if(result ~= cheetah_mask(i, j))
                error = error + 1;
            end

        end
end

% display my mask
imshow(maskRes)
title("MAP Method")

% result (error)
errorRate = error / totalPixels;
disp(errorRate)
```

Code for Plotting Graphs

```
clear
clc
clf

% —— Plot Graphs ——

% load results
load("Alpha.mat")
load("errorMAP_D4P2.mat")
load("errorPE_D4P2.mat")
load("errorML_D4.mat")

% set up x-axis and y-axes
x = alpha;

```

```
y1 = errorBayes_D4P2;
y2 = errorML_D4;
y3 = errorMAP_D4P2;

% plot the graph
semilogx(x, y1, x, y2, x, y3)
title("Data 4 Strategy 2")
legend('PE', 'ML', 'MAP')
grid on
```