The plots are shown below:

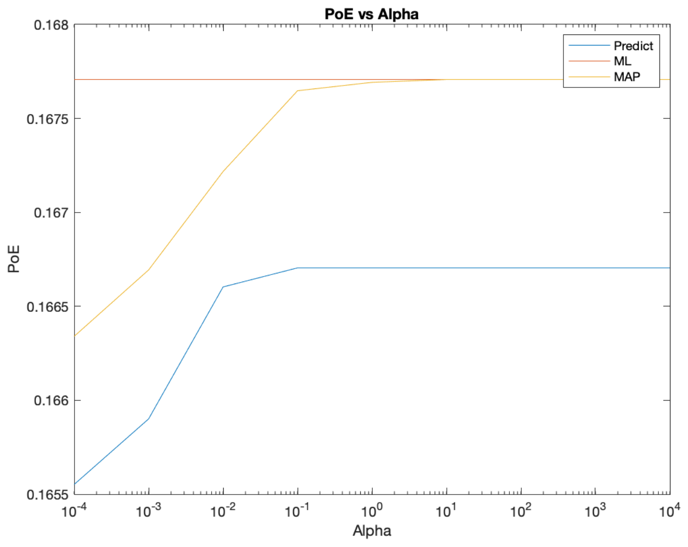

fig.1 Dataset 1-Strategy 1

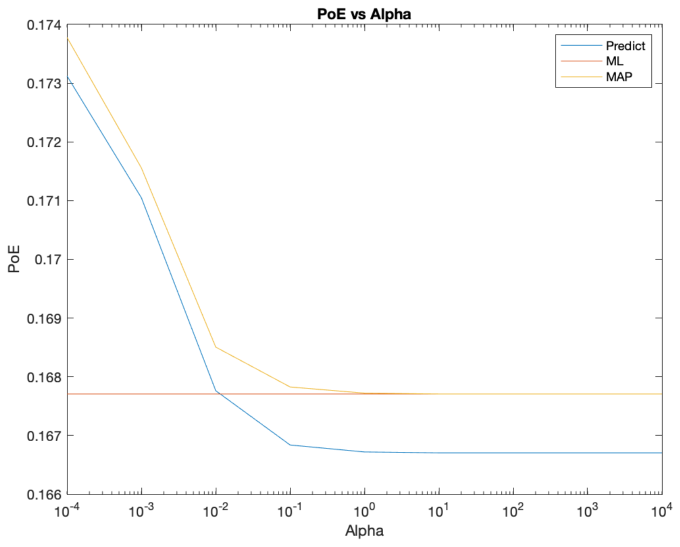

fig.2 Dataset 1-Strategy 2

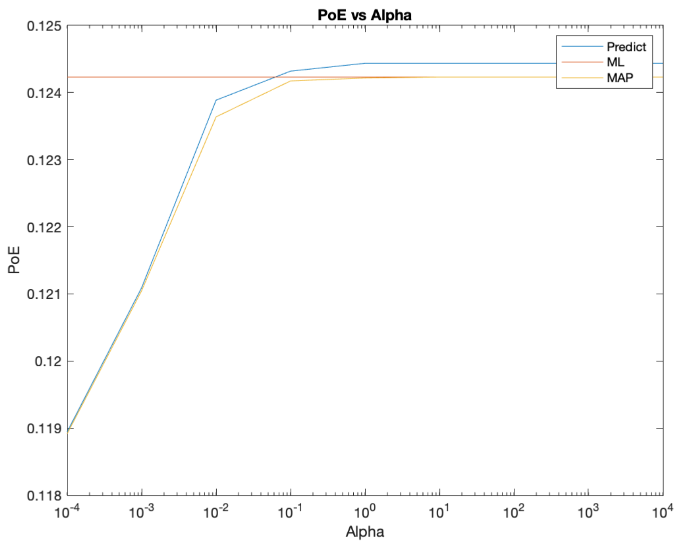

fig.3 Dataset 2-Strategy 1

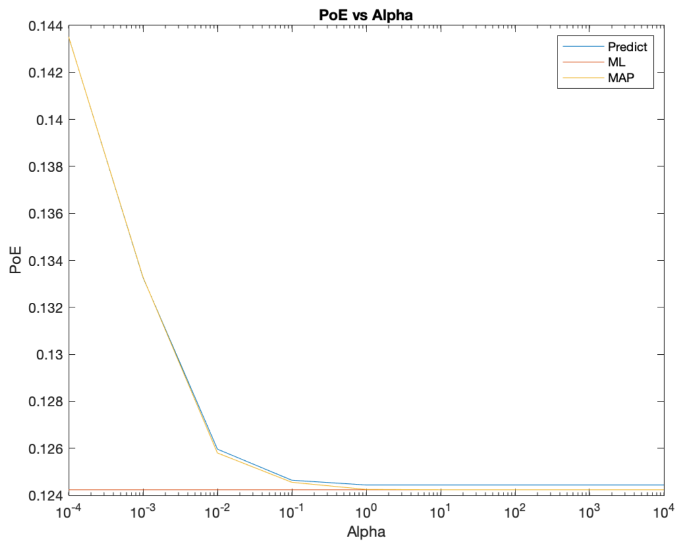

fig.4 Dataset 2-Strategy 2

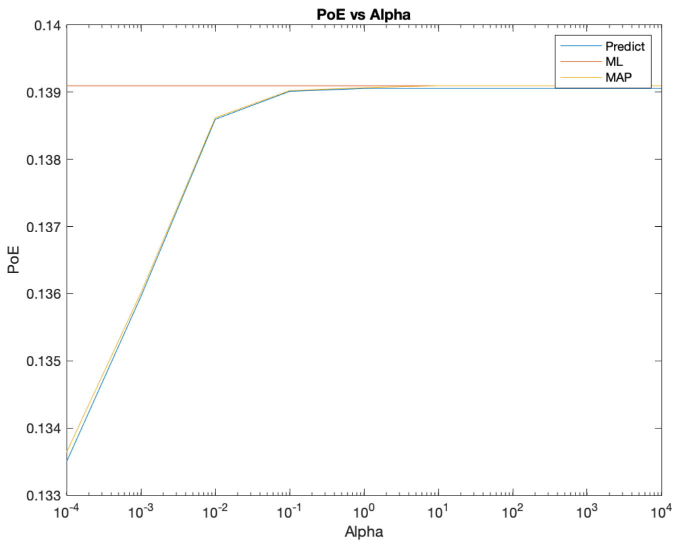

fig.5 Dataset 3 -Strategy 1

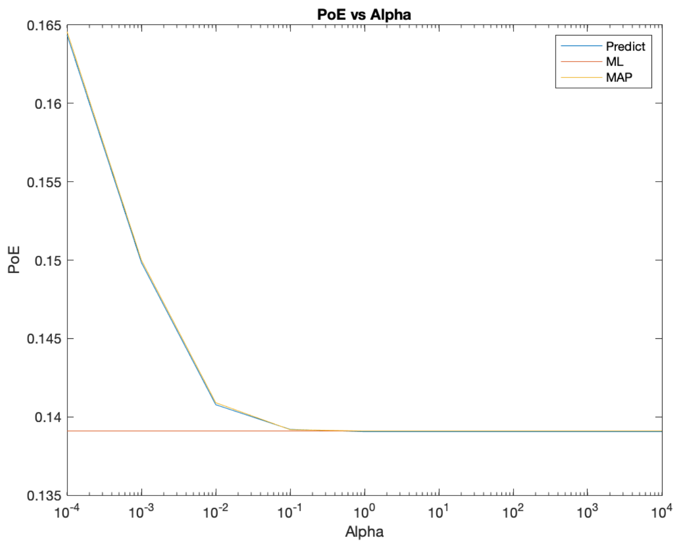

fig.6 Dataset 3 -Strategy 2

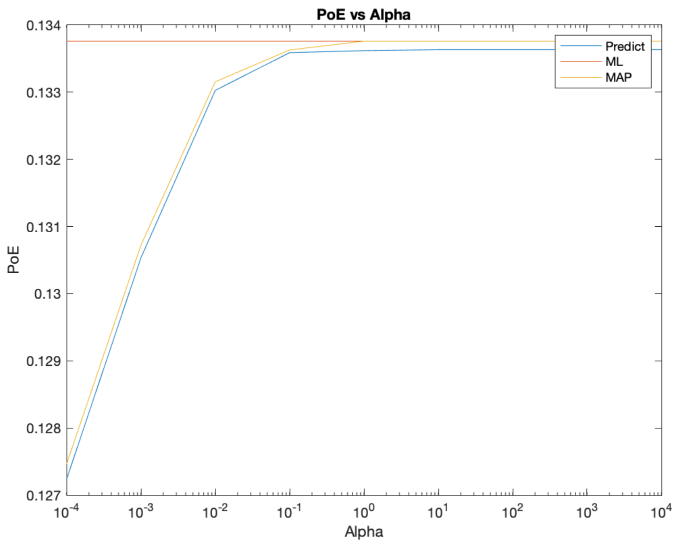

fig.7 Dataset 4-Strategy 1

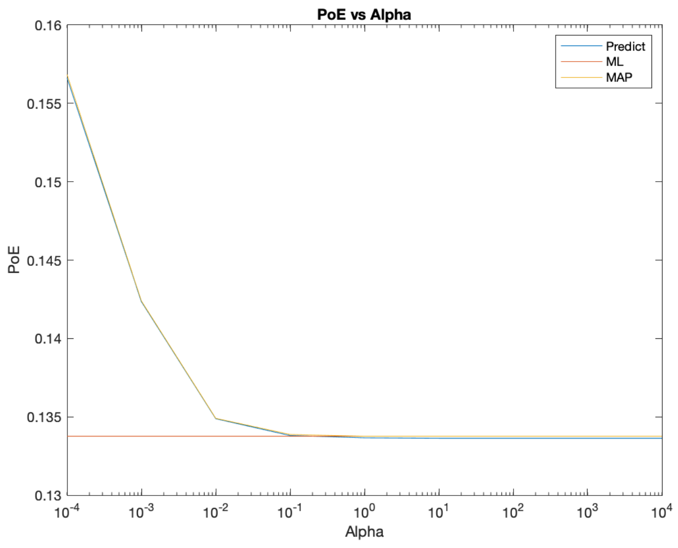

fig.8 Dataset 4-Strategy 2

$$\mu_n = \frac{n\sigma_0^2}{\sigma^2 + n\sigma_0^2}\mu_{ML} + \frac{\sigma^2}{\sigma^2 + n\sigma_0^2}\mu_0$$

1) the relative behavior of these three curves

For Dataset 1 with strategy 1, we can observe that with the Alpha increasing, both the PoE (probability of error) of the Predict and MAP are increasing, while the PoE of ML remains constant. Finally, with the Alpha increasing, the PoE of MAP tends to approach PoE of ML at around 0.1677. However, PoE of Predict approaches at around 0.1666, which is smaller than the other two results.

Because ML does not depend on prior, so PoE of ML always remains constant. When the Alpha is relatively small, due to the intuitive combination of mean value, we weigh more on the prior assumption. With the Alpha increasing, observation plays a more important role in the mean value. That is why the PoE of Predict and MAP are trying to approach the PoE of ML when Alpha becomes larger. Though both the PoE of Predict and MAP increases with larger Alpha, the final result of Predict is better than the of MAP. It is because Dataset 1 have relatively small size and proper prior assumption. In this case, the relatively large different mean value among them results in different PoE.

2) how that behavior changes from dataset to dataset

For all the Datasets with strategy 1, with the Alpha increasing, all the PoE converge to a constant value finally. In Dataset 1, the PoE of Predict is significantly smaller than ML and MAP compared with the other three datasets. While in Datasets 2, 3 and 4, all three curves converge to a very close error value (little deviation though).

We have already known that the Datasets have different sizes. The size is the primary reason that cause the different results. We can find that Dataset 1 has the smallest data size, the prior have the largest influence on the result, resulting in lower PoE of Predict due to the proper prior assumption. In other Datasets, since the size increases, we weigh more on the observation than the prior. So when Alpha becomes large enough, the PoE of three curves converges to a similar constant value.

3) how all of the above change when strategy 1 is replaced by strategy 2.

When we try to change from strategy 1 to strategy 2, the most obvious change is that the PoE of Predict and MAP will decrease with the Alpha increasing. In both strategies, PoE of ML always remains a constant value.

It is obvious that the mean μ_0 should be different between cheetah class and grass class. When applying strategy 1, we can observe that the choices of mean μ_0 have better behavior with smaller Alpha. Because when the Alpha is relatively small, we weigh more on the prior than the observation. In this case, we can say the prior assumption of strategy 1 is closer to the real distribution. The result shows that the mean μ_0 of strategy 2 is worse than strategy 1. The reason why the PoE of Predict and MAP decrease with Alpha increasing is because we weigh more on the observation than the prior when Alpha is relatively larger.

To draw a conclusion, different mean μ_0 assumption will have different impact on the behaviors. With Alpha increasing, we weigh more on observation, and thus reduces the impact of prior assumption.

Bolin He, PID: A53316428, Hw03

Nov 27,2019

```
clear all;
close all;
clc;

Dataset = load('TrainingSamplesDCT_subsets_8.mat');
Alpha = load('Alpha.mat');
Alpha = Alpha.alpha;

for strategy = 1:2

    if strategy == 1
        strategy = load('Prior_1.mat');
    elseif strategy == 2
        strategy = load('Prior_2.mat');
    end

    for dataset = 1:4
        if dataset == 1
            BG = Dataset.D1_BG;
            FG = Dataset.D1_FG;
        elseif dataset == 2
            BG = Dataset.D2_BG;
            FG = Dataset.D2_FG;
        elseif dataset == 3
            BG = Dataset.D3_BG;
            FG = Dataset.D3_FG;
        elseif dataset == 4
            BG = Dataset.D4_BG;
            FG = Dataset.D4_FG;
        end

        bayes_error = [];
        mle_error = [];
        map_error = [];
        n_FG = length(FG);
        n_BG = length(BG);

        % Initialization
        for alpha_idx = 1:length(Alpha)
            cov_0 = zeros(64,64);
            for idx = 1:64
               cov_0(idx,idx) = Alpha(alpha_idx)*strategy.W0(idx);
            end

            % FG
            FG_cov = cov(FG);
            t = inv(cov_0 + (1/n_FG)*FG_cov);
            mu_1_FG = cov_0 * t * transpose(mean(FG)) + (1/n_FG) *
FG_cov * t * transpose(strategy.mu0_FG);
            cov_1_FG = cov_0 * t * (1/n_FG) * FG_cov;

            % predictive distribution of FG
            mu_pred_FG = mu_1_FG;
            cov_pred_FG = FG_cov + cov_1_FG;

            % BG
            BG_cov = cov(BG);
            t2 = inv(cov_0 + (1/n_BG)*BG_cov);
            mu_1_BG = cov_0 * t2 * transpose(mean(BG)) + (1/n_BG) *
BG_cov * t2 * transpose(strategy.mu0_BG);
            cov_1_BG = cov_0 * t2 * (1/n_BG) * BG_cov;

            % predictive distribution of BG
            mu_pred_BG = mu_1_BG;
            cov_pred_BG = BG_cov + cov_1_BG;

            % Prior
            num_FG = length(FG);
            num_BG = length(BG);
            prior_FG = num_FG / (num_FG + num_BG);
            prior_BG = num_BG / (num_FG + num_BG);

            % Bayes
            I = imread('cheetah.bmp');
            I = im2double(I);

            % Paddling
            I = [zeros(size(I,1),2) I];
            I = [zeros(2, size(I,2)); I];
            I = [I zeros(size(I,1),5)];
            I = [I; zeros(5, size(I,2))];

            % DCT
            [m,n] = size(I);
            Blocks = ones(m-7,n-7);
            det_cov_FG = det(cov_pred_FG);
            det_cov_BG = det(cov_pred_BG);
            ave_tmp_FG = transpose(mu_pred_FG);
            ave_tmp_BG = transpose(mu_pred_BG);
            inv_tmp_FG = inv(cov_pred_FG);
            inv_tmp_BG = inv(cov_pred_BG);

            % predict
            const_FG = ave_tmp_FG*inv_tmp_FG*transpose(ave_tmp_FG) +
log(det_cov_FG) - 2*log(prior_FG);
            const_BG = ave_tmp_BG*inv_tmp_BG*transpose(ave_tmp_BG) +
log(det_cov_BG) - 2*log(prior_BG);

            for i=1:m-7
                for j=1:n-7
                    DCT = dct2(I(i:i+7,j:j+7));
                    zigzag_order = zigzag(DCT);
                    feature = zigzag_order;
                    g_cheetah = 0;
                    g_grass = 0;
                    % cheetah
                    g_cheetah = g_cheetah +
feature*inv_tmp_FG*transpose(feature);
                    g_cheetah = g_cheetah -
2*feature*inv_tmp_FG*transpose(ave_tmp_FG);
                    g_cheetah = g_cheetah + const_FG;
                    % grass
                    g_grass = g_grass +
feature*inv_tmp_BG*transpose(feature);
                    g_grass = g_grass -
2*feature*inv_tmp_BG*transpose(ave_tmp_BG);
                    g_grass = g_grass + const_BG;
                    if g_cheetah >= g_grass
                        Blocks(i,j) = 0;
                    end
                end
            end

            % save
            ground_truth = imread('cheetah_mask.bmp')/255;
            [x,y] = size(ground_truth);
            count1 = 0;
            count2 = 0;
            count_cheetah_truth = 0;
            count_grass_truth = 0;
            for i=1:x
                for j=1:y
                    if prediction(i,j) > ground_truth(i,j)
                        count2 = count2 + 1;
                        count_grass_truth = count_grass_truth + 1;
                    elseif prediction(i,j) < ground_truth(i,j)
                        count1 = count1 + 1;
                        count_cheetah_truth = count_cheetah_truth + 1;
                    elseif ground_truth(i,j) >0
                        count_cheetah_truth = count_cheetah_truth + 1;
                    else
                        count_grass_truth = count_grass_truth + 1;
                    end
                end
            end
            error1_64 = (count1/count_cheetah_truth) * prior_FG;
            error2_64 = (count2/count_grass_truth) * prior_BG;
            total_error_64 = error1_64 + error2_64;
            bayes_error = [bayes_error total_error_64];
            % ML
            % DCT
            [m,n] = size(I);
            Blocks = ones(m-7,n-7);
            mean_FG = mean(FG);
            mean_BG = mean(BG);
            ave_tmp_FG = mean_FG;
            ave_tmp_BG = mean_BG;
            inv_covFG = inv(FG_cov);
            inv_covBG = inv(BG_cov);
            DcovFG = det(FG_cov);
            DcovBG = det(BG_cov);

            % predict
            const_FG = ave_tmp_FG*inv_covFG*transpose(ave_tmp_FG) +
log(DcovFG) - 2*log(prior_FG);
            const_BG = ave_tmp_BG*inv_covBG*transpose(ave_tmp_BG) +
log(DcovBG) - 2*log(prior_BG);

            for i=1:m-7
                for j=1:n-7
                    DCT = dct2(I(i:i+7,j:j+7));
                    zigzag_order = zigzag(DCT);
                    feature = zigzag_order;
                    g_cheetah = 0;
                    g_grass = 0;
                    % cheetah
                    g_cheetah = g_cheetah +
feature*inv_covFG*transpose(feature);
                    g_cheetah = g_cheetah -
2*feature*inv_covFG*transpose(ave_tmp_FG);
                    g_cheetah = g_cheetah + const_FG;
                    % grass
                    g_grass = g_grass +
feature*inv_covBG*transpose(feature);
                    g_grass = g_grass -
2*feature*inv_covBG*transpose(ave_tmp_BG);
                    g_grass = g_grass + const_BG;
                    if g_cheetah >= g_grass
                        Blocks(i,j) = 0;
                    end
                end
            end

            % save
            count1 = 0;
            count2 = 0;
            count_cheetah_truth = 0;
            count_grass_truth = 0;
            for i=1:x
                for j=1:y
                    if prediction(i,j) > ground_truth(i,j)
                        count2 = count2 + 1;
                        count_grass_truth = count_grass_truth + 1;
                    elseif prediction(i,j) < ground_truth(i,j)
                        count1 = count1 + 1;
                        count_cheetah_truth = count_cheetah_truth + 1;
                    elseif ground_truth(i,j) >0
                        count_cheetah_truth = count_cheetah_truth + 1;
                    else
                    count_grass_truth = count_grass_truth + 1;
                end
            end
        end
        error1_64 = (count1/count_cheetah_truth) * prior_FG;
        error2_64 = (count2/count_grass_truth) * prior_BG;
        total_error_64 = error1_64 + error2_64;
        mle_error = [mle_error total_error_64];

        % MAP
        % DCT
        [m,n] = size(I);
        Blocks = ones(m-7,n-7);
        det_cov_FG = det(FG_cov);
        det_cov_BG = det(BG_cov);
        ave_tmp_FG = transpose(mu_pred_FG);
        ave_tmp_BG = transpose(mu_pred_BG);

        % predict
        const_FG = ave_tmp_FG*inv_covFG*transpose(ave_tmp_FG) +
log(det_cov_FG) - 2*log(prior_FG);
        const_BG = ave_tmp_BG*inv_covBG*transpose(ave_tmp_BG) +
log(det_cov_BG) - 2*log(prior_BG);

        for i=1:m-7
            for j=1:n-7
                DCT = dct2(I(i:i+7,j:j+7));
                zigzag_order = zigzag(DCT);
                feature = zigzag_order;
                g_cheetah = 0;
                g_grass = 0;
                % cheetah
                g_cheetah = g_cheetah +
feature*inv_covFG*transpose(feature);
                g_cheetah = g_cheetah -
2*feature*inv_covFG*transpose(ave_tmp_FG);
                g_cheetah = g_cheetah + const_FG;
                % grass
                g_grass = g_grass +
feature*inv_covBG*transpose(feature);
                g_grass = g_grass -
2*feature*inv_covBG*transpose(ave_tmp_BG);
                g_grass = g_grass + const_BG;
                if g_cheetah >= g_grass
                    Blocks(i,j) = 0;
                end
            end
        end

        % save
        count1 = 0;
        count2 = 0;
        count_cheetah_truth = 0;
        count_grass_truth = 0;
```

```
                for i=1:x
                    for j=1:y
                        if prediction(i,j) > ground_truth(i,j)
                            count2 = count2 + 1;
                            count_grass_truth = count_grass_truth + 1;
                        elseif prediction(i,j) < ground_truth(i,j)
                            count1 = count1 + 1;
                            count_cheetah_truth = count_cheetah_truth + 1;
                        elseif ground_truth(i,j) >0
                            count_cheetah_truth = count_cheetah_truth + 1;
                        else
                            count_grass_truth = count_grass_truth + 1;
                        end
                    end
                end
                error1_64 = (count1/count_cheetah_truth) * prior_FG;
                error2_64 = (count2/count_grass_truth) * prior_BG;
                total_error_64 = error1_64 + error2_64;
                map_error = [map_error total_error_64];
            end

            % plot
            plot(Alpha,bayes_error,Alpha,mle_error,Alpha,map_error);
            legend('Predict','ML','MAP');
            set(gca, 'XScale', 'log');
            xlabel('Alpha');
            ylabel('PoE');
            title('PoE vs Alpha');
            saveas(gcf,['Strategy_' int2str(strategy) '_dataset_'
 int2str(dataset) '_PoEvsAlpha.png']);
    end
end
```

```
Undefined function 'zigzag' for input arguments of type 'double'.

Error in ECE271_hw3_he (line 101)
                    zigzag_order = zigzag(DCT);
```

Published with MATLAB® R2018a