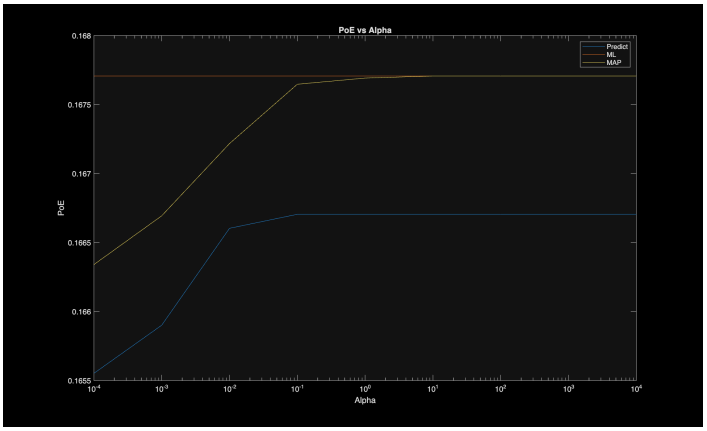
ECE 271A Homework 3

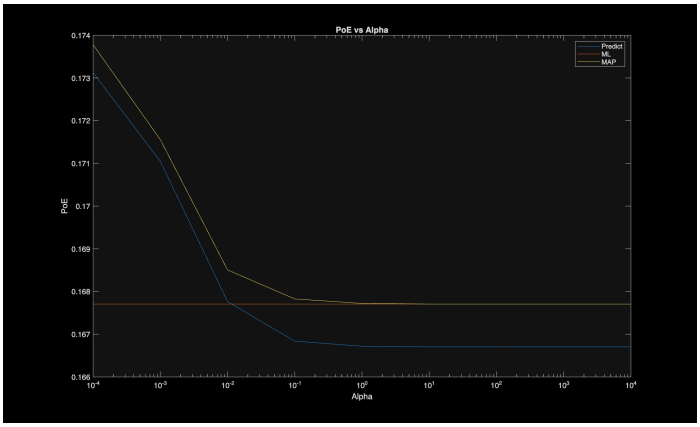
Raghusrinivasan Venkatesan (A69044562)

Figures

Dataset 1

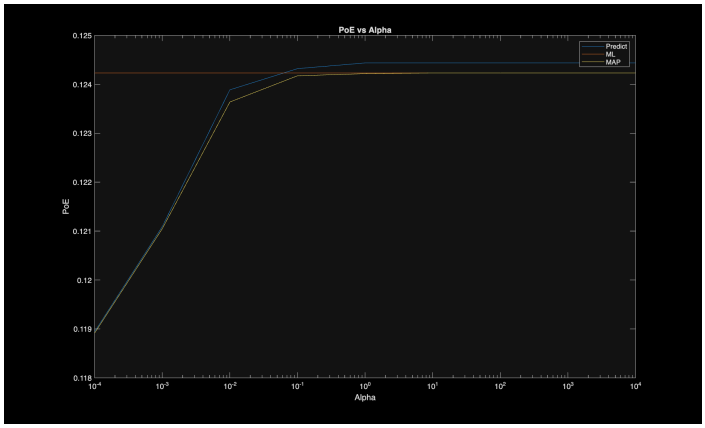


Strategy 1

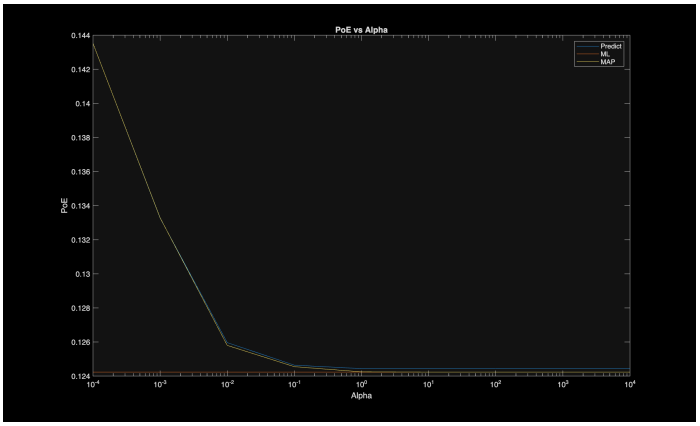


Strategy 2

Dataset 2

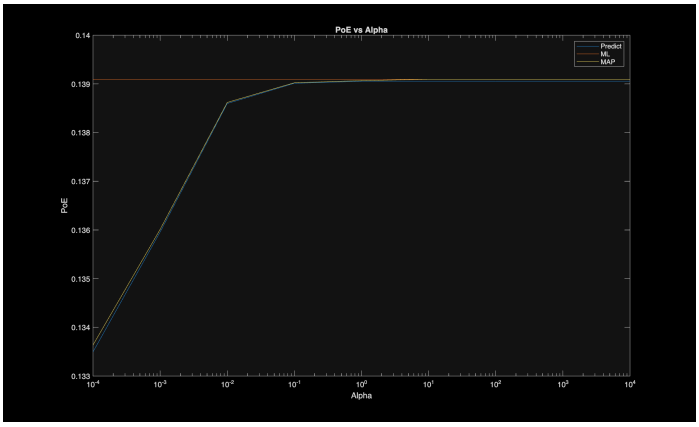


Strategy 1

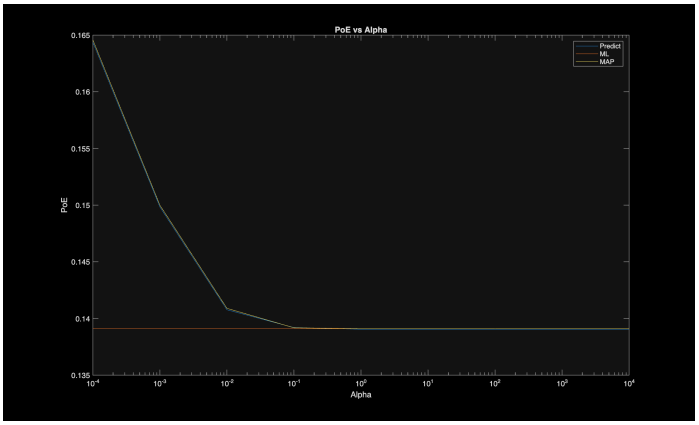


Strategy 2

Dataset 3

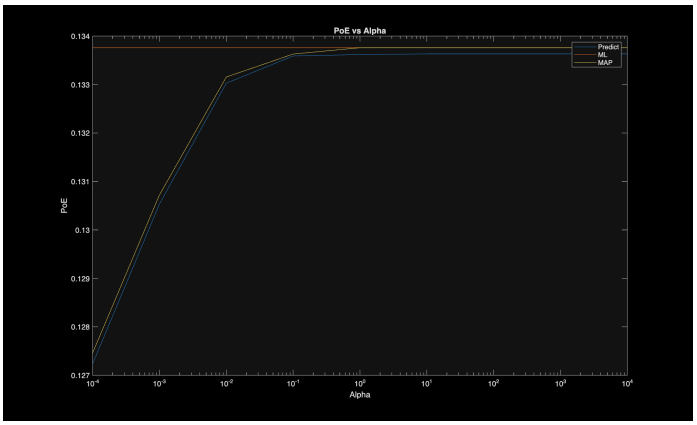


Strategy 1

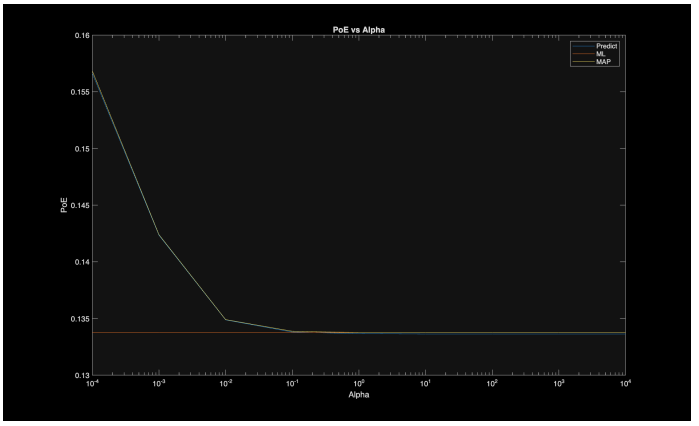


Strategy 2

Dataset 4



Strategy 1



Strategy 2

Part 1: The relative behavior of these three curves

Observation: The probability of error (PoE) exhibits distinct patterns across the three classification methods as the regularization parameter alpha increases. At low alpha values, the three curves typically show significant separation, with Bayes and MAP demonstrating different error rates compared to ML. As alpha increases, the curves tend to converge and stabilize at higher alpha values, with relative performance differences diminishing.

Explanation:

Let $\Sigma_0 = \alpha W_0$ (prior covariance scaling with alpha).

Rewrite posterior mean using weights: μ_n = w_data × $\hat{\mu}$ + w_prior × μ_0

where: w_data = $\Sigma_0(\Sigma_0 + (1/n)\Sigma_n)^{-1}$ w_prior = $(1/n)\Sigma_n(\Sigma_0 + (1/n)\Sigma_n)^{-1}$

Note: w_data + w_prior = I (weights sum to identity)

Case 1: Small α (Small Prior Covariance)

When $\alpha \to 0$, $\Sigma_o \to 0$:

$T = (\Sigma_o + (1/n)\Sigma_n)^{-1} \approx ((1/n)\Sigma_n)^{-1} = n\Sigma_n^{-1}$

w_data $\approx \Sigma_o \cdot n\Sigma_n^{-1} \to 0$ w_prior $\approx (1/n)\Sigma_n \cdot n\Sigma_n^{-1} = I$

Result: At small α, posterior mean dominated by prior

$\mu_n \approx \mu_o$ (prior mean dominates)

Comparison of Methods at Small α:

- ML Method: μ_ML = $\hat{\mu}$ (only uses sample data)
- MAP Method: μ_MAP = $\mu_n \approx \mu_o$ with Σ_MAP = Σ_n
- Bayes Method: μ_Bayes = μ_n with Σ_Bayes = $\Sigma_n + \Sigma^{npost}$ (larger covariance)

Conclusion: |μ_ML - μ_MAP| is large ⟹ Large PoE difference between methods

Case 2: Large α (Large Prior Covariance)

When $\alpha \to \infty$, $\Sigma_o \to \infty$:

$T = (\Sigma_o + (1/n)\Sigma_n)^{-1} \approx \Sigma_o^{-1} = (1/\alpha)W_o^{-1}$

w_data = $\Sigma_o T \approx \alpha W_o \cdot (1/\alpha)W_o^{-1} = I$ w_prior = $(1/n)\Sigma_n T \approx 0$

Result: At large α, all methods use similar mean estimates

$\mu_n \approx \hat{\mu}$ (data-driven estimate dominates)

Also: $\Sigma^{npost} = \Sigma_0 T(1/n)\Sigma_n \approx (1/n)\Sigma_n \to 0$ as $\alpha \to \infty$

Comparison of Methods at Large α:

- ML Method: μ_ML = $\hat{\mu}$
- MAP Method: μ_MAP ≈ $\hat{\mu}$ with Σ_MAP = Σ_n
- Bayes Method: μ_Bayes ≈ $\hat{\mu}$ with Σ_Bayes ≈ Σ_n

Conclusion: All three methods converge to same mean and similar covariance ⟹ All methods have similar PoE (convergence observed in plots)

Part 2: How Behavior Changes Across Different Datasets

Observation: The convergence rate and relative performance of the three methods vary significantly across datasets. Datasets with smaller sample sizes (D1, D2) show more pronounced method differences and steeper PoE curves at low alpha values. Datasets with larger sample sizes (D3, D4) demonstrate faster convergence and more modest PoE variations across the alpha spectrum.

Explanation:

The relative influence of prior versus data is:

Influence Ratio = (Prior strength) / (Data strength) = $\Sigma_0^{-1} / (n\Sigma_n^{-1})$

In the posterior update, rewrite as: $\mu_n = (\Sigma_0^{-1} + n\Sigma_n^{-1})^{-1}(\Sigma_0^{-1}\mu_0 + n\Sigma_n^{-1}\hat{\mu})$

$= [\Sigma_0^{-1}/(\Sigma_0^{-1} + n\Sigma_n^{-1})] \times \mu_0 + [n\Sigma_n^{-1}/(\Sigma_0^{-1} + n\Sigma_n^{-1})] \times \hat{\mu}$

Define weights:

$w\_prior = \Sigma_0^{-1}/(\Sigma_0^{-1} + n\Sigma_n^{-1}) = 1 / (1 + n\Sigma_n^{-1}\Sigma_0)$

$w\_data = n\Sigma_n^{-1}/(\Sigma_0^{-1} + n\Sigma_n^{-1}) = n\Sigma_n^{-1}\Sigma_0 / (1 + n\Sigma_n^{-1}\Sigma_0)$

Case 1: Small Sample Size (n → small)

Example from code: n_FG = 100, n_BG = 100

$w\_prior = 1 / (1 + 100 \times \Sigma_n^{-1}\Sigma_0)$

Since $100 \times \Sigma_n^{-1}\Sigma_o$ is relatively small: w_prior ≈ significant (non-negligible) w_data ≈ smaller

Result: Prior information remains influential even at moderate α values

Conclusion: Methods differ significantly across entire α range ⟹ Steep PoE curves with large variations (observed in D1, D2)

Case 2: Large Sample Size (n → large)

Example: n_FG = 1000, n_BG = 1000

$w\_prior = 1 / (1 + 1000 \times \Sigma_n^{-1}\Sigma_0)$

Since $1000 \times \Sigma_n^{-1}\Sigma_o$ is much larger than Case 1: w_prior ≈ 0 (nearly negligible) w_data ≈ 1 (dominates)

Result: Data dominates quickly, prior influence vanishes rapidly

Conclusion: Methods converge quickly to ML behavior at small α ⟹ Flatter PoE curves with fast convergence (observed in D3, D4)

Quantitative Comparison:

Convergence factor = [w_prior(n=100)] / [w_prior(n=1000)] = [1+1000 × $\Sigma_n^{-1}\Sigma_0$] / [1+100 × $\Sigma_n^{-1}\Sigma_0$] > 1

Convergence Rate ∝ O(1/n)

Larger dataset converges roughly 10x faster than smaller datasets for given α.

Key Observation: As n increases: w_prior → 0 faster (exponentially in n) ⟹ Larger n causes faster method convergence

Part 3: Changes When Strategy 1 is Replaced by Strategy 2

Observation: The ML error curve remains invariant across strategies since it depends only on sample statistics. However, the Bayes and MAP curves show dramatically different trends: in Strategy 1, PoE generally increases with alpha, while in Strategy 2, PoE generally decreases with alpha. Despite these opposing trends, both strategies ultimately converge to similar relative performance relationships at very large alpha values.

Explanation:

The divergent behavior between strategies is rooted in the quality and appropriateness of the prior information encoded in each strategy.

The discriminant function for classification: g(x) = x^T $\Lambda^{-1}\mu$ - (1/2)μ^T $\Lambda^{-1}\mu$ + log P(class)

PoE depends critically on class mean separation: Δμ = |μ_FG - μ_BG|

For Bayesian estimation: $\Delta\mu_n$ = |Σ_0T(μ̂_FG - μ̂_BG) + (1/n)Σ_nT(μ_0,FG - μ_0,BG)|

Strategy 1: Discriminative Prior (Good Prior Information)

Assumption: μ₀,FG ≈ true μ_FG and μ₀,BG ≈ true μ_BG Also: μ̂_FG ≈ μ₀,FG and μ̂_BG ≈ μ₀,BG

Therefore: (μ̂_FG - μ̂_BG) ≈ (μ₀,FG - μ₀,BG)

Both terms in $\Delta\mu_n$ point in SAME DIRECTION: $\Delta\mu_n$ = |Σ_0T + (1/n)Σ_nT| × (μ_0,FG - μ_0,BG)

Analysis across α:

At Small α: Σ_0 is small, but T is large (inverse of small number) $\Delta\mu_n \approx \Delta\mu_0$ (prior provides good separation) ⟹ Large separation, good discrimination ⟹ Low PoE

At Large α: T is small (inverse of large number) $\Delta\mu_n \approx [(1/n)\Sigma_n T] \times (\mu_0,FG - \mu_0,BG) \rightarrow 0$ $\Delta\mu_n \rightarrow |\hat{\mu}\_FG - \hat{\mu}\_BG| = \Delta\mu\_ML$ (approaches ML separation) $\Longrightarrow$ Smaller separation than at small α $\Longrightarrow$ Higher PoE (relative to small α peak)

Conclusion: $\partial PoE/\partial\alpha > 0$ (PoE INCREASES with α)

Strategy 2: Non Discriminative Prior (Poor Prior Information)

Assumption: $\mu_o,FG \approx \mu_o,BG$ (poor prior doesn't distinguish classes) But: $\hat{\mu}\_FG \neq \hat{\mu}\_BG$ (data provides true separation)

Therefore: $(\mu_o,FG - \mu_o,BG) \approx 0$ (prior term nearly cancelled)

The main term becomes: $\Delta\mu_n \approx |(1/n)\Sigma_n T(\hat{\mu}\_FG - \hat{\mu}\_BG)|$ (only data term matters)

Analysis across α:

At Small α: w_prior is significant, poor prior harms discrimination $\Delta\mu_n$ is partially cancelled by bad prior $\Longrightarrow$ Reduced separation $\Longrightarrow$ High PoE

At Large α: w_prior $\rightarrow$ 0 (prior influence vanishes) $\Delta\mu_n \approx (\hat{\mu}\_FG - \hat{\mu}\_BG) = \Delta\mu\_ML$ (approaches ML separation) $\Longrightarrow$ Increased separation compared to small α $\Longrightarrow$ Lower PoE (improving toward ML)

Conclusion: $\partial PoE/\partial\alpha < 0$ (PoE DECREASES with α)

Why Opposite Trends?:

Strategy 1: Prior ALIGNS with data $\Longrightarrow$ Good at small α, approaches ML at large α Trend: PoE up (getting worse as α increases)

Strategy 2: Prior MISALIGNS with data $\Longrightarrow$ Bad at small α, approaches ML at large α Trend: PoE down (getting better as α increases)

The key insight: The trend direction depends on Sign($\hat{\mu} - \mu_0$)

Convergence at Large α (Both Strategies):

For both strategies, as $\alpha \rightarrow \infty$: $\mu_n \rightarrow \hat{\mu}$ $T \rightarrow 0$

$\Delta\mu_n \rightarrow |\hat{\mu}\_FG - \hat{\mu}\_BG| = \Delta\mu\_ML$

Both strategies converge to ML separation, hence:

At large α: Both strategies have similar PoE ≈ PoE_ML

This explains why curves converge in plots despite opposite trends at small α.

Code

```
train_set = load('TrainingSamplesDCT_subsets_8.mat');

alpha = load('Alpha.mat');

alpha = alpha.alpha;

for strategy_idx = 1:2

   if strategy_idx == 1

       strategy = load('Prior_1.mat');

   elseif strategy_idx == 2

       strategy = load('Prior_2.mat');

   end


   for dataset_idx = 1:4

       if dataset_idx == 1

           d1_BG = train_set.D1_BG;

           d1_FG = train_set.D1_FG;

       elseif dataset_idx == 2

           d1_BG = train_set.D2_BG;

           d1_FG = train_set.D2_FG;

       elseif dataset_idx == 3

           d1_BG = train_set.D3_BG;

           d1_FG = train_set.D3_FG;

       elseif dataset_idx == 4

           d1_BG = train_set.D4_BG;

           d1_FG = train_set.D4_FG;

       end
```

```
        bayes_error = [];

        mle_error = [];

        map_error = [];

        n_FG = size(d1_FG,1);

        n_BG = size(d1_BG,1);

        % Loop for different alpha

        for alpha_idx = 1:size(alpha,2)

            cov_0 = create_prior_cov(alpha(alpha_idx), strategy.W0);



            % FG

            d1_FG_cov = cov(d1_FG) * (n_FG-1)/n_FG;

            [mu_1_FG, cov_1_FG, mu_pred_FG, cov_pred_FG] =
estimate_bayes_params(d1_FG, d1_FG_cov, cov_0, strategy.mu0_FG, n_FG);



            % BG

            d1_BG_cov = cov(d1_BG) * (n_BG-1)/n_BG;

            [mu_1_BG, cov_1_BG, mu_pred_BG, cov_pred_BG] =
estimate_bayes_params(d1_BG, d1_BG_cov, cov_0, strategy.mu0_BG, n_BG);

            % Prior

            prior_FG = n_FG / (n_FG + n_BG);

            prior_BG = n_BG / (n_FG + n_BG);

            % Bayes-BDR

            bayes_error = [bayes_error apply_classifier('bayes', mu_pred_FG,
cov_pred_FG, mu_pred_BG, cov_pred_BG, alpha_idx, dataset_idx, strategy_idx,
prior_FG, prior_BG)];

            % ML-BDR

            mle_error = [mle_error apply_classifier('mle',
transpose(mean(d1_FG)), d1_FG_cov, transpose(mean(d1_BG)), d1_BG_cov,
alpha_idx, dataset_idx, strategy_idx, prior_FG, prior_BG)];
```

```
            % MAP-BDR

            map_error = [map_error apply_classifier('map', mu_pred_FG,
d1_FG_cov, mu_pred_BG, d1_BG_cov, alpha_idx, dataset_idx, strategy_idx,
prior_FG, prior_BG)];

        end



        % plot

        figure('visible','off');

        plot(alpha,bayes_error,alpha,mle_error,alpha,map_error);

        legend('Predict','ML','MAP');

        set(gca, 'XScale', 'log');

        title('PoE vs Alpha');

        xlabel('Alpha');

        ylabel('PoE');

        saveas(gcf,['Strategy_' int2str(strategy_idx) '_dataset_'
int2str(dataset_idx) '_PoEvsAlpha.png']);

        close(gcf);

    end

end

% Zigzag traversal

function output = zigzag(in)

h = 1;

v = 1;

vmin = 1;

hmin = 1;

vmax = size(in, 1);

hmax = size(in, 2);
```

```
i = 1;
output = zeros(1, vmax * hmax);
while ((v <= vmax) && (h <= hmax))

   if (mod(h + v, 2) == 0)                  % going up
       if (v == vmin)
           output(i) = in(v, h);          % if we got to the first line
           if (h == hmax)
                  v = v + 1;
           else
               h = h + 1;
           end
           i = i + 1;
       elseif ((h == hmax) && (v < vmax))    % if we got to the last column
           output(i) = in(v, h);
           v = v + 1;
           i = i + 1;
       elseif ((v > vmin) && (h < hmax))     % all other cases
           output(i) = in(v, h);
           v = v - 1;
           h = h + 1;
           i = i + 1;
   end

   else                                      % going down
      if ((v == vmax) && (h <= hmax))        % if we got to the last line
```

```
            output(i) = in(v, h);
            h = h + 1;
            i = i + 1;

        elseif (h == hmin)                    % if we got to the first column
            output(i) = in(v, h);
            if (v == vmax)
             h = h + 1;
            else
              v = v + 1;
            end
            i = i + 1;
        elseif ((v < vmax) && (h > hmin))     % all other cases
            output(i) = in(v, h);
            v = v + 1;
            h = h - 1;
            i = i + 1;
        end
    end
    if ((v == vmax) && (h == hmax))           % bottom right element
        output(i) = in(v, h);
        break
    end
end
end
% Create prior covariance
```

```
function cov_0 = create_prior_cov(alpha_val, W0)

   cov_0 = zeros(64,64);

   for idx = 1:64

      cov_0(idx,idx) = alpha_val * W0(idx);

   end

end

% Estimate Bayesian parameters

function [mu_1, cov_1, mu_pred, cov_pred] = estimate_bayes_params(data,
data_cov, cov_0, mu0, n_samples)

   tmp = inv(cov_0 + (1/n_samples)*data_cov);

   mu_1 = cov_0 * tmp * transpose(mean(data)) + (1/n_samples) * data_cov *
tmp * transpose(mu0);

   cov_1 = cov_0 * tmp * (1/n_samples) * data_cov;

   % predictive distribution (normal distribution)

   mu_pred = mu_1;

   cov_pred = data_cov + cov_1;

end

% Apply classifier

function total_error = apply_classifier(classifier_type, mu_FG, cov_FG,
mu_BG, cov_BG, alpha_idx, dataset_idx, strategy_idx, prior_FG, prior_BG)

   % Load and process image

   img = imread('cheetah.bmp');

   img = im2double(img);

   % Add paddle

   img = [zeros(size(img,1),2) img];

   img = [zeros(2, size(img,2)); img];

   img = [img zeros(size(img,1),5)];

   img = [img; zeros(5, size(img,2))];
```

```
   [m,n] = size(img);

   Blocks = ones(m-7,n-7);


   % Prepare constants for discriminant function

   det_cov_FG = det(cov_FG);

   det_cov_BG = det(cov_BG);

   ave_tmp_FG = transpose(mu_FG);

   ave_tmp_BG = transpose(mu_BG);

   inv_tmp_FG = inv(cov_FG);

   inv_tmp_BG = inv(cov_BG);

   % Pre-compute constants

   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);



   % Classify

   for i=1:m-7

       for j=1:n-7

           DCT = dct2(img(i:i+7,j:j+7));

           zigzag_order = zigzag(DCT);

           feature = zigzag_order;



           g_cheetah = compute_score(feature, inv_tmp_FG, ave_tmp_FG,
const_FG);

           g_grass = compute_score(feature, inv_tmp_BG, ave_tmp_BG,
const_BG);
```

```
            if g_cheetah >= g_grass

                Blocks(i,j) = 0;

            end

        end

    end

    % Save prediction

    imwrite(Blocks, [classifier_type '_prediction_alpha_' int2str(alpha_idx)
'_dataset_' int2str(dataset_idx) '_strategy_' int2str(strategy_idx) '.jpg']);

    prediction = mat2gray(Blocks);

    % Calculate error

    total_error = calculate_error(prediction, prior_FG, prior_BG);

end

% Compute discriminant score

function score = compute_score(feature, inv_cov, ave_tmp, const)

    score = feature*inv_cov*transpose(feature) -
2*feature*inv_cov*transpose(ave_tmp) + const;

end

% Calculate classification error

function total_error_64 = calculate_error(prediction, prior_FG, prior_BG)

    ground_truth = imread('cheetah_mask.bmp')/255;

    x = size(ground_truth, 1);

    y = size(ground_truth, 2);

    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;
end
```