ECE 271A - Statistical Learning I - Assignment 5

Saqib Azim (A59010162)
UC San Diego
sazim@ucsd.edu

Dec 3, 2021

1 Segmentation of Cheetah Image into cheetah (foreground) and ground (background)

Figure 1: Original Cheetah Image

Figure 2: GT Segmentation Mask

1.1 Probability of Error Formula

$$P(E) = E_Y(P_{X|Y}(g(x) \neq i)|i) \tag{1}$$

$$P(E) = P_{X|Y}(g(x) \neq \text{FG}|\text{FG})P_Y(\text{FG}) + P_{X|Y}(g(x) \neq \text{BG}|\text{BG})P_Y(\text{BG}) \tag{2}$$

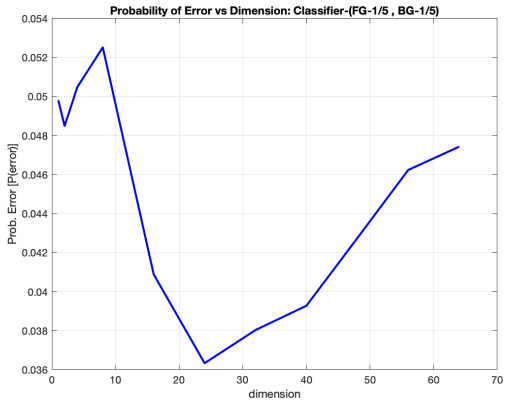

Figure 3: Probability of Error VS Dimension - (FG,BG) - (1,1)

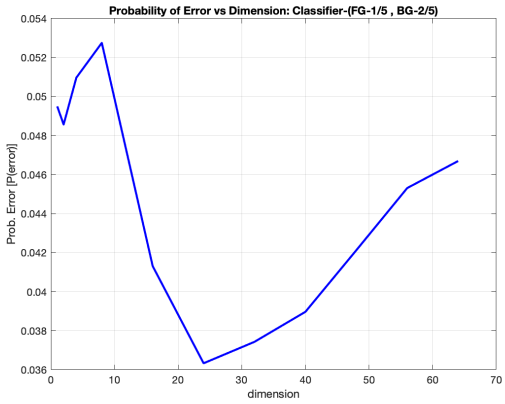

Figure 4: Probability of Error VS Dimension - (FG,BG) - (1,2)

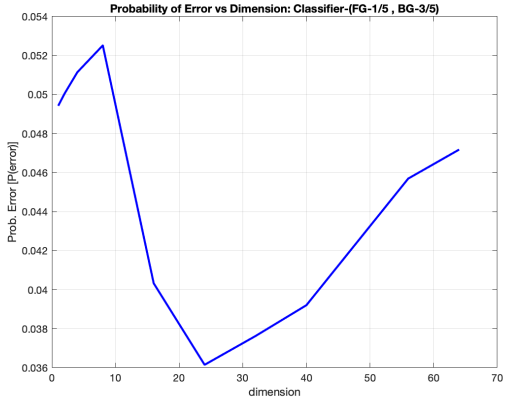

Figure 5: Probability of Error VS Dimension - (FG,BG) - (1,3)

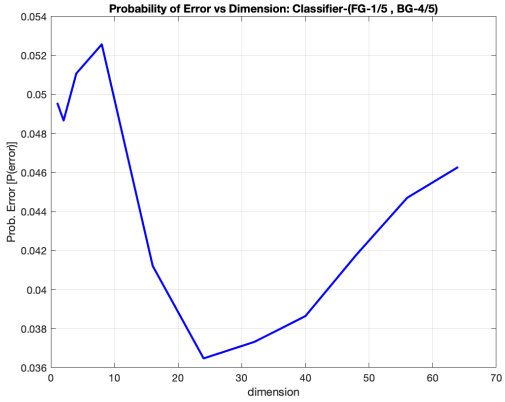

Figure 6: Probability of Error VS Dimension - (FG,BG) - (1,4)

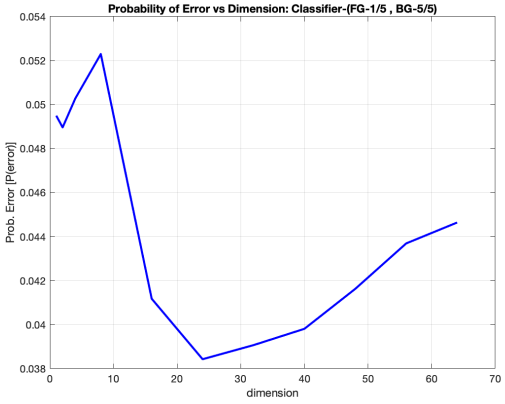

Figure 7: Probability of Error VS Dimension - (FG,BG) - (1,5)

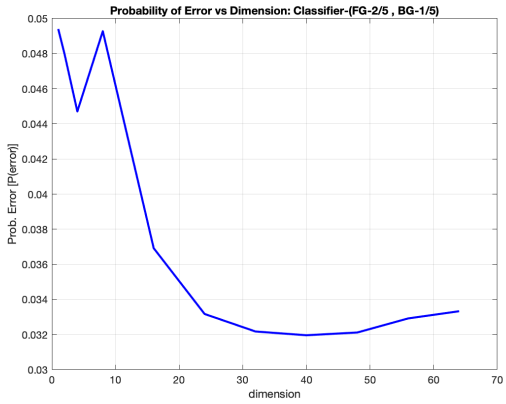

Figure 8: Probability of Error VS Dimension - (FG,BG) - (2,1)

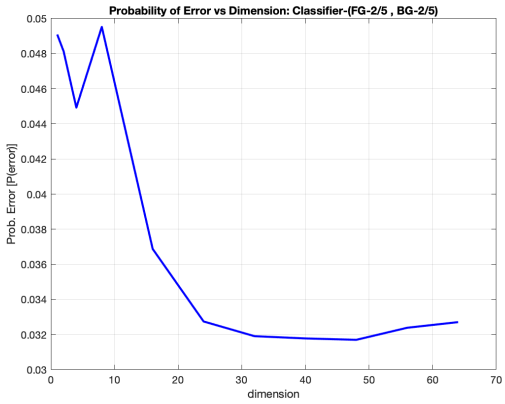

Figure 9: Probability of Error VS Dimension - (FG,BG) - (2,2)

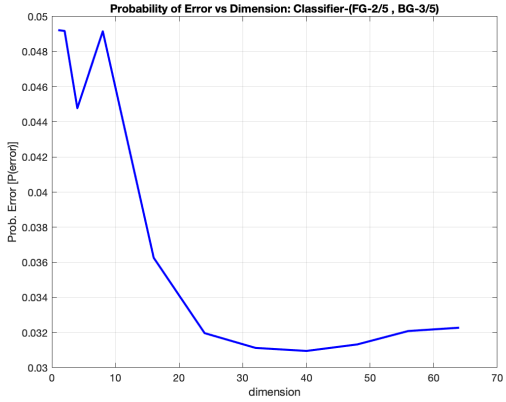

Figure 10: Probability of Error VS Dimension - (FG,BG) - (2,3)

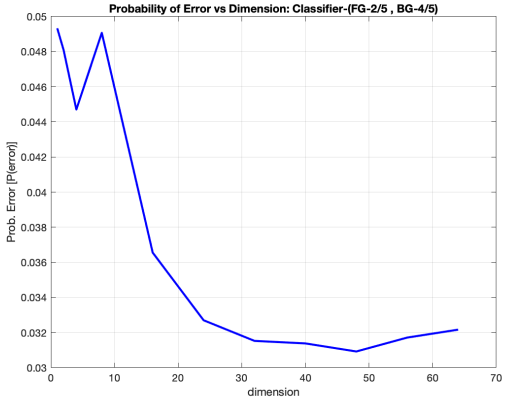

Figure 11: Probability of Error VS Dimension - (FG,BG) - (2,4)

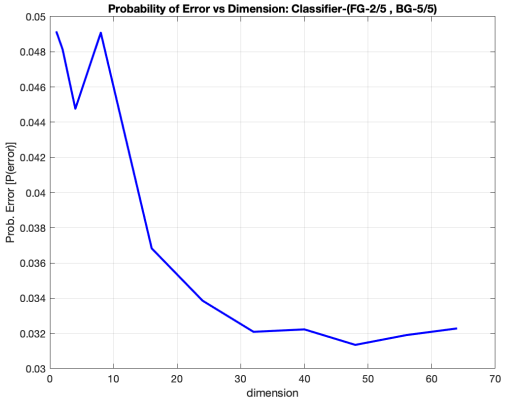

Figure 12: Probability of Error VS Dimension - (FG,BG) - (2,5)

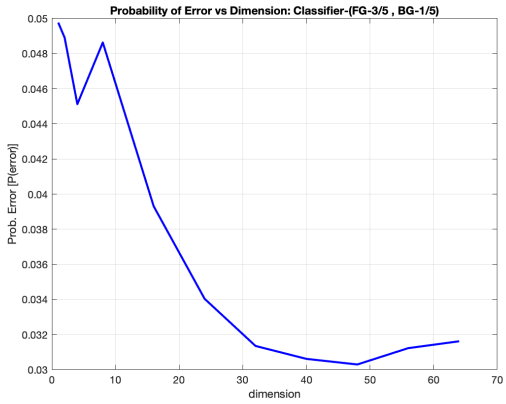

Figure 13: Probability of Error VS Dimension - (FG,BG) - (3,1)

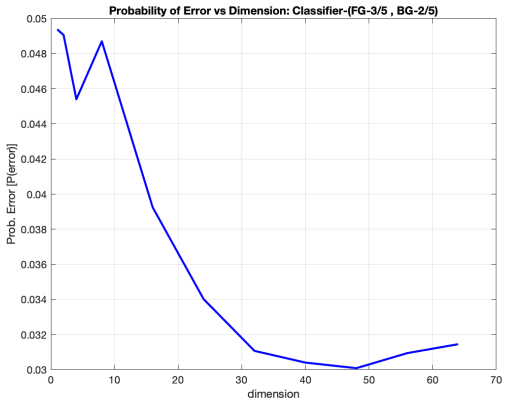

Figure 14: Probability of Error VS Dimension - (FG,BG) - (3,2)

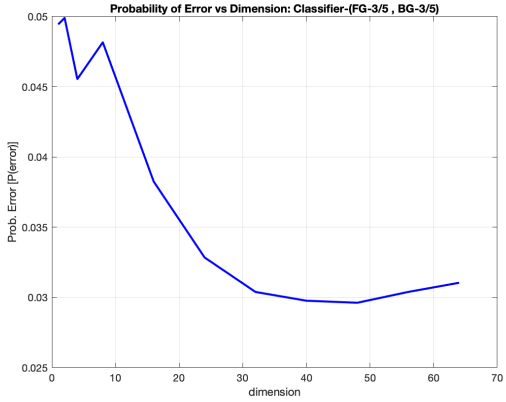

Figure 15: Probability of Error VS Dimension - (FG,BG) - (3,3)

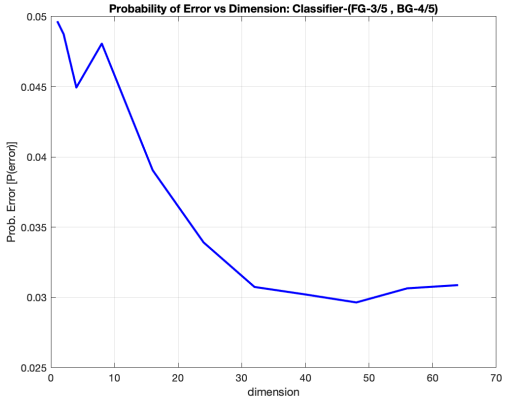

Figure 16: Probability of Error VS Dimension - (FG,BG) - (3,4)

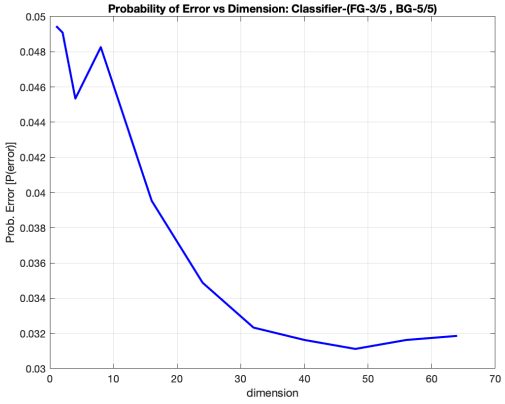

Figure 17: Probability of Error VS Dimension - (FG,BG) - (3,5)

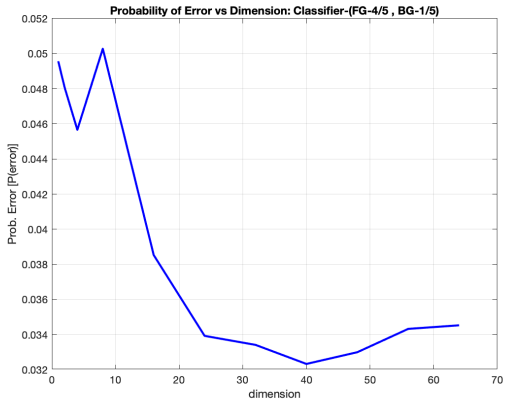

Figure 18: Probability of Error VS Dimension - (FG,BG) - (4,1)

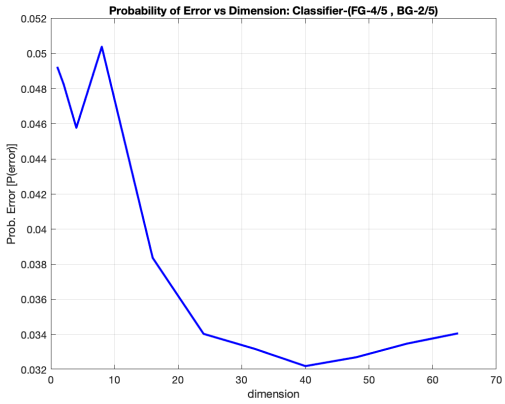

Figure 19: Probability of Error VS Dimension - (FG,BG) - (4,2)

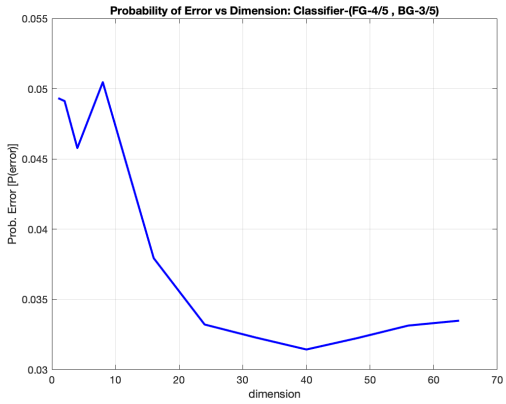

Figure 20: Probability of Error VS Dimension - (FG,BG) - (4,3)

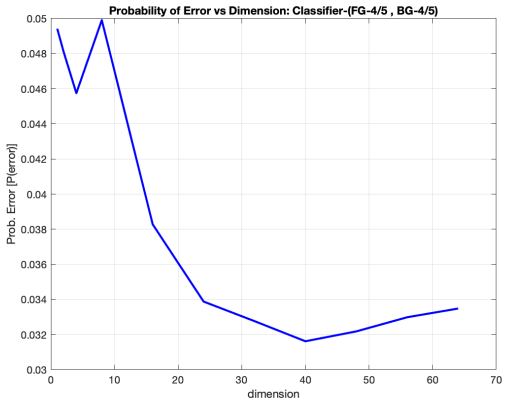

Figure 21: Probability of Error VS Dimension - (FG,BG) - (4,4)

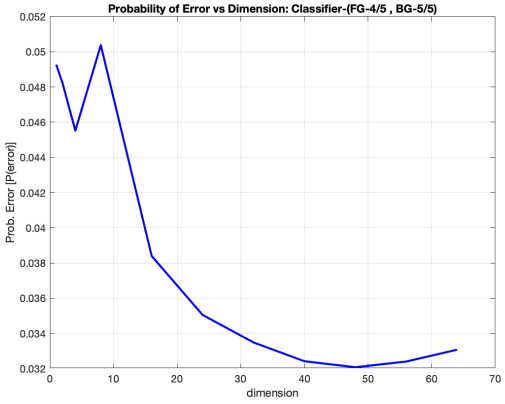

Figure 22: Probability of Error VS Dimension - (FG,BG) - (4,5)

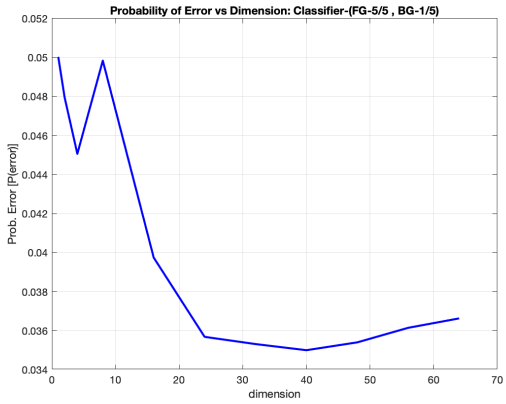

Figure 23: Probability of Error VS Dimension - (FG,BG) - (5,1)

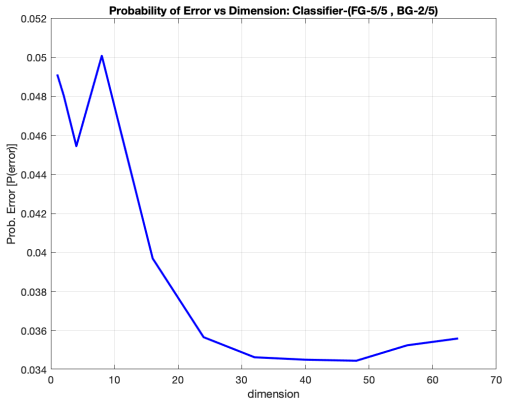

Figure 24: Probability of Error VS Dimension - (FG,BG) - (5,2)

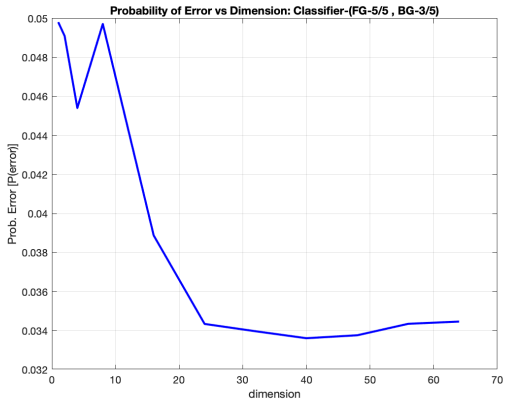

Figure 25: Probability of Error VS Dimension - (FG,BG) - (5,3)

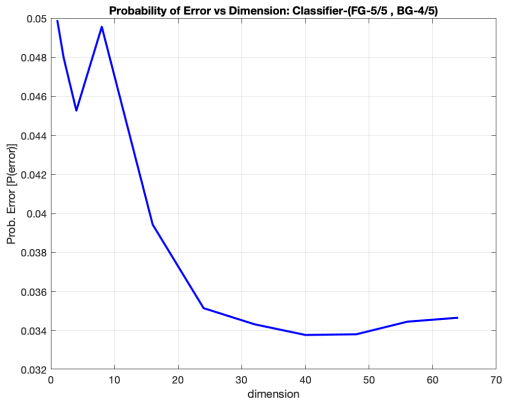

Figure 26: Probability of Error VS Dimension - (FG,BG) - (5,4)

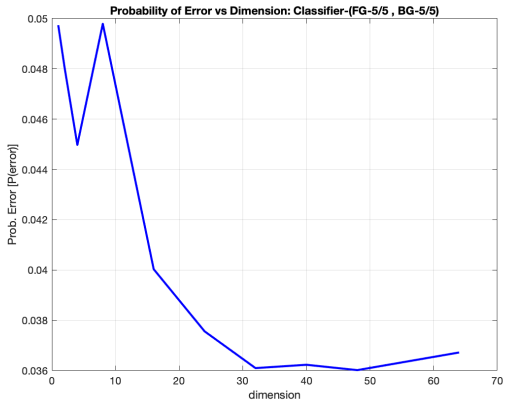

Figure 27: Probability of Error VS Dimension - (FG,BG) - (5,5)

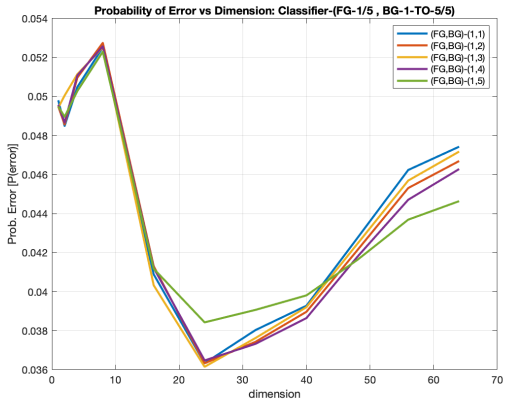

Figure 28: Probability of Error VS Dimension - (FG,BG) - (1,1-5)

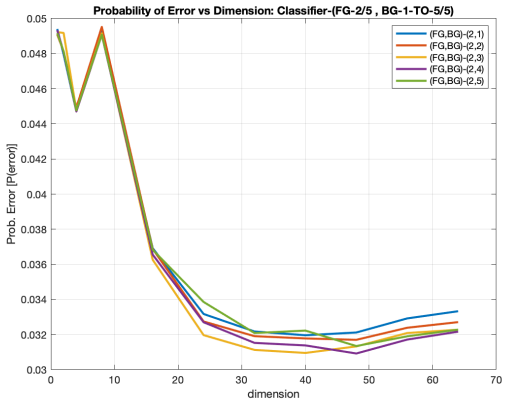

Figure 29: Probability of Error VS Dimension - (FG,BG) - (2,1-5)

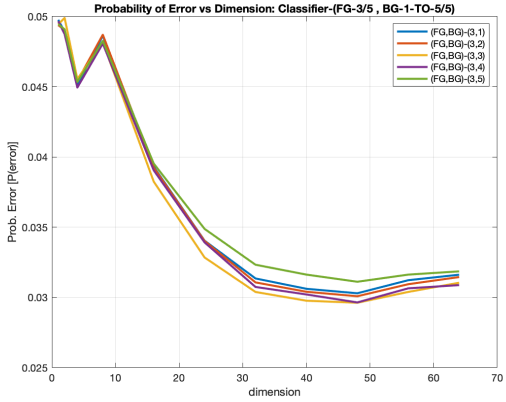

Figure 30: Probability of Error VS Dimension - (FG,BG) - (3,1-5)

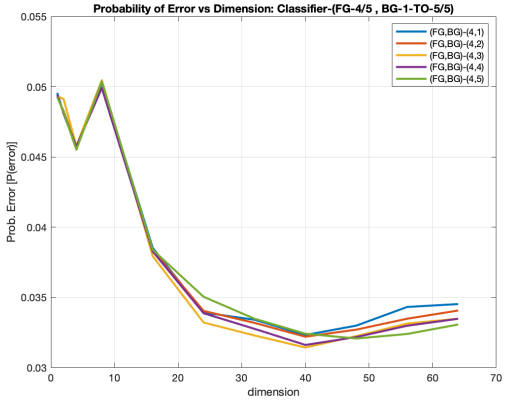

Figure 31: Probability of Error VS Dimension - (FG,BG) - (4,1-5)

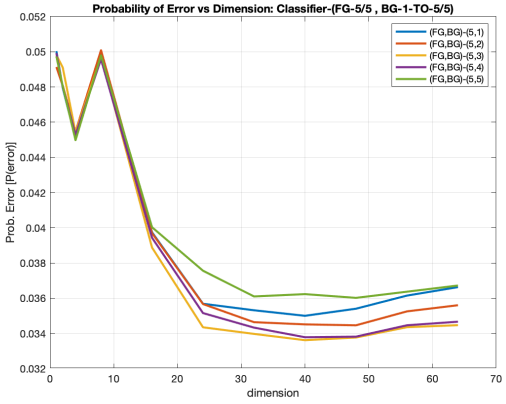

Figure 32: Probability of Error VS Dimension - (FG,BG) - (5,1-5)

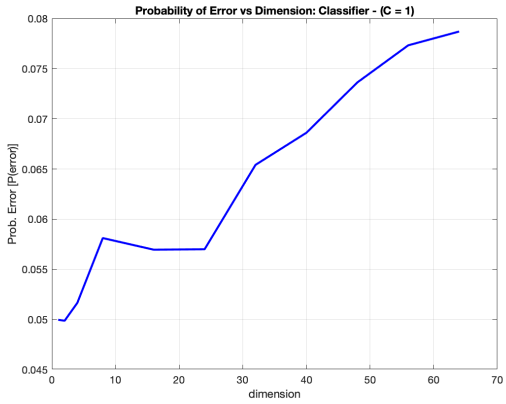

Figure 33: Probability of Error VS Dimension - C = 1

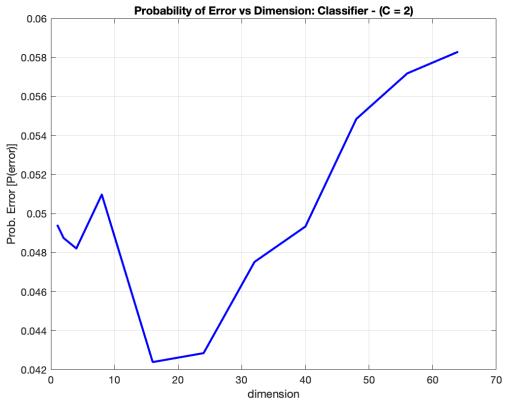

Figure 34: Probability of Error VS Dimension - C = 2

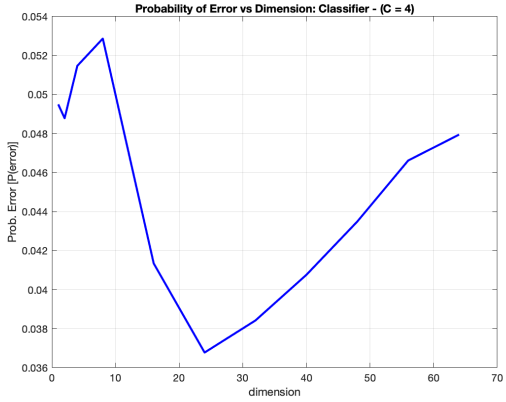

Figure 35: Probability of Error VS Dimension - C = 4

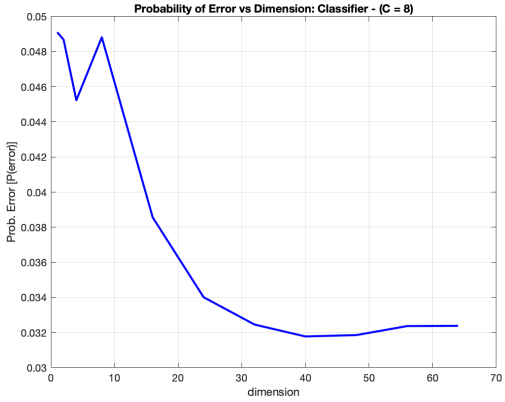

Figure 36: Probability of Error VS Dimension - C = 8

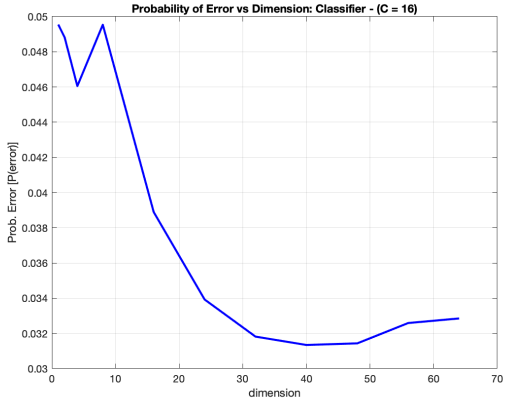

Figure 37: Probability of Error VS Dimension - C = 16

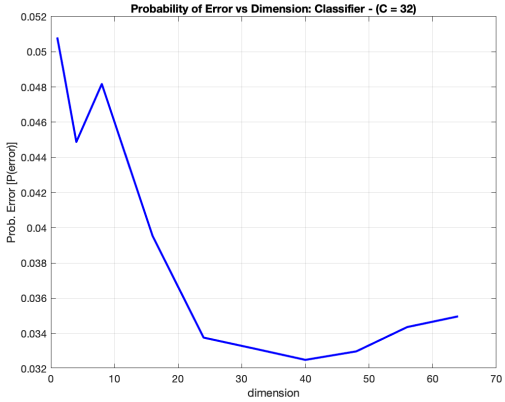

Figure 38: Probability of Error VS Dimension - C = 32

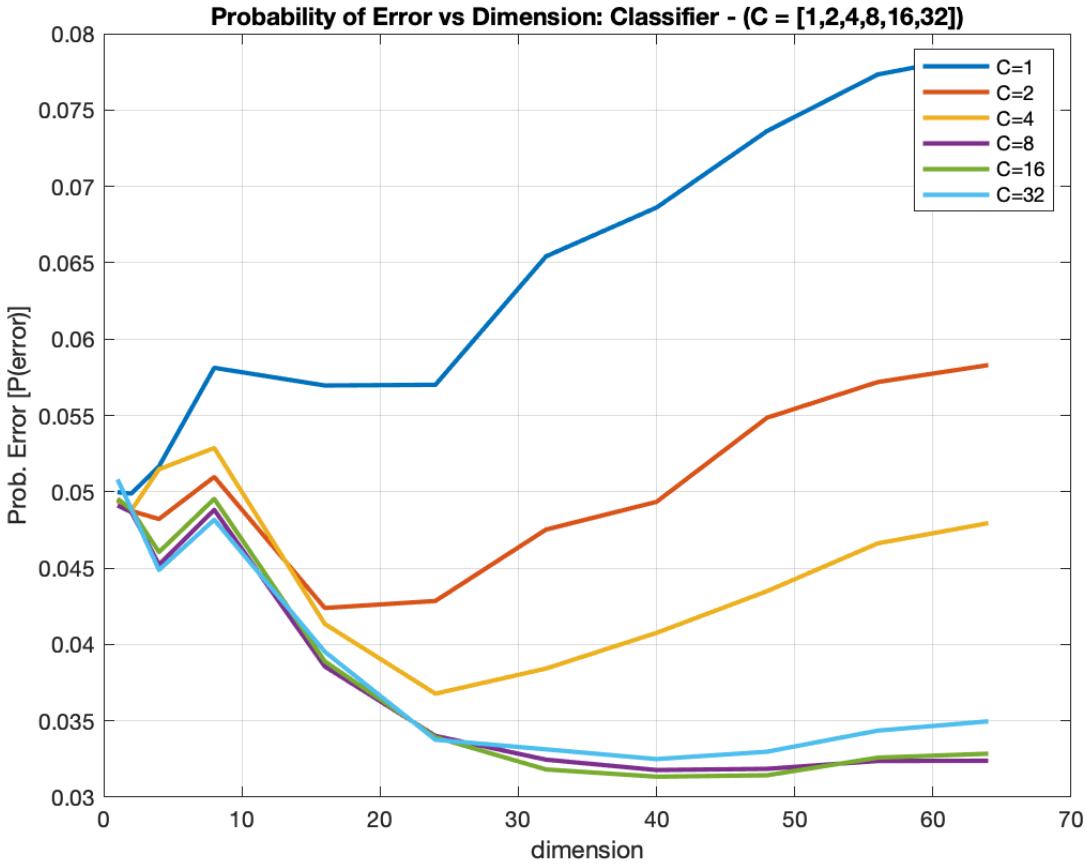

Figure 39: Probability of Error VS Dimension - C = [1,2,4,8,16,32]

1.2 Comment: Dependence of Probability of Error on Initialization

Following observations can be made from Fig. 28 - 32:

- Each plot shows variation of error probability wrt dimension for a single mixture model learned for FG class combined with 5 mixture models learned for BG class
- Initialization is important as a bad random initialization can lead to varying results. For example, in Fig. 28, the error probability increases with dimension after certain point. This is different from Fig. 29-32 which can be attributed to bad parameter initialization in the EM algorithm for the first mixture model learned for foreground (cheetah) class
- For lower dimensions ($d \leq 10$), there is a small fluctuation in the curve for all the 25 plots irrespective of the initialization
- For all parameter initialization, the error probability decreases and becomes more or less stable as the dimensions increase

- For higher dimensions ($d \geq 50$), the error probability curve slightly increases

1.3 Effect of number of mixture components on probability of error

Following observations can be made from Fig. 39:

- For small feature dimensions ($d \leq 10$), the probability of error takes similar values for C = 1,2,4,8,16,32
- For all values of C, the curve fluctuates for smaller dimensions and becomes stable for higher feature dimensions
- As dimensions increase, plots with lower number of mixture components (lower C value) have higher probability of error than plots with large number of mixture components
- As the number of mixture component increases (C increases), the prob. error vs dimension plot goes lower
- For large values of C (=8,16,32), the curve is nearly identical. This indicates that after a certain point, adding more mixture models may be redundant and may be useless in modeling the data distribution
- For small values of C (=1,2,4), the curve fluctuates for small feature dimensions, and increases as the dimension increase
- For large values of C (=8,16,32), the curve fluctuates for small feature dimensions, but decreases as the dimension increase

2 Appendix

2.1 Matlab Code - Main Code: 'main_Qa.m'

```
clc; clear;
close all;

% load the original image and ground-truth segmentation mask
img = im2double(imread("../data/cheetah.bmp"));
img = img(:, 1:end-2);
seg_mask_gt = im2double(imread('../data/cheetah_mask.bmp'));
seg_mask_gt = seg_mask_gt(:, 1:end-2);

% load the zigzag pattern file
zigzag_pat = importdata("../data/zigzag_pattern.txt");
zigzag_pat_lin = zigzag_pat(:) + 1;   % adding 1 for converting to matlab indexes

[img_height, img_width] = size(img);
```

```
% pad test image with 7 layers to the right and bottom
img_pad = img_padding(img);

% compute the dct of the padded image once and reuse it for all test
dct_dim = 64;                           % feature dimension
dct_mat = zeros(img_height * img_width, dct_dim);
itr = 0;

for i = 1:img_height
    for j = 1:img_width
        itr = itr + 1;
        img_block_dct = dct2(img_pad(i:i+7, j:j+7));
        dct_mat(itr, zigzag_pat_lin) = img_block_dct(:);
    end
end

disp("Image DCT Computed !!!");

% load the training sample DCT matrix
TS_DCT = load("../data/TrainingSamplesDCT_8_new.mat");
TS_DCT_FG = TS_DCT.TrainsampleDCT_FG;
TS_DCT_BG = TS_DCT.TrainsampleDCT_BG;

% compute the prior class probabilities
num_sample_FG = size(TS_DCT_FG, 1);
num_sample_BG = size(TS_DCT_BG, 1);
num_sample = num_sample_FG + num_sample_BG;

PY_FG = num_sample_FG / num_sample;
PY_BG = num_sample_BG / num_sample;

C = 8;   % number of components
d = 64;     % feature size/dimension

max_itr = 5;

mu_FG_mat = zeros(C, d, max_itr);
mu_BG_mat = zeros(C, d, max_itr);
pi_FG_mat = zeros(C, 1, max_itr);
pi_BG_mat = zeros(C, 1, max_itr);
sigma_FG_mat = zeros(d, d, C, max_itr);
sigma_BG_mat = zeros(d, d, C, max_itr);

for itr = 1:max_itr
```

```
    [mu_FG_mat(:,:,itr), sigma_FG_mat(:,:,:,itr), pi_FG_mat(:,:,itr)] = EM_algorithm(TS_
    [mu_BG_mat(:,:,itr), sigma_BG_mat(:,:,:,itr), pi_BG_mat(:,:,itr)] = EM_algorithm(TS_

    disp("EM - Parameters Estimation Complete - "+itr+"/"+max_itr);
end

dim_list = [1,2,4,8,16,24,32,40,48,56,64];
num_dim = size(dim_list, 2);
prob_error = zeros(max_itr * max_itr, num_dim);

top_itr = 0;

for itr_FG = 1:max_itr
    mu_FG = mu_FG_mat(:,:,itr_FG);
    sigma_FG = sigma_FG_mat(:,:,:,itr_FG);
    pi_FG = pi_FG_mat(:,:,itr_FG);

    for itr_BG = 1:max_itr
        mu_BG = mu_BG_mat(:,:,itr_BG);
        sigma_BG = sigma_BG_mat(:,:,:,itr_BG);
        pi_BG = pi_BG_mat(:,:,itr_BG);

        top_itr = top_itr + 1;

        disp("Classifier - ("+itr_FG+"/"+max_itr+" , "+itr_BG+"/"+max_itr+")");

        for idx = 1:num_dim
            d = dim_list(1, idx);
            seg_mask_res = zeros(size(img));

            % compute gaussian formula for speed
            num_test_samples = size(dct_mat, 1);
            PXGY_x_FG = zeros(num_test_samples, 1);
            PXGY_x_BG = zeros(num_test_samples, 1);

            for k = 1:C
                term1_FG = 1 / sqrt(power(2 * pi, d) * det(sigma_FG(1:d,1:d,k)));
                x_minus_mu_FG = dct_mat(:,1:d) - mu_FG(k,1:d);
                sigma_FG_inv = inv(sigma_FG(1:d,1:d,k));
                PXGY_x_FG = PXGY_x_FG + term1_FG * exp(-0.5 * sum((x_minus_mu_FG * sigma

                term1_BG = 1 / sqrt(power(2 * pi, d) * det(sigma_BG(1:d,1:d,k)));
                x_minus_mu_BG = dct_mat(:,1:d) - mu_BG(k,1:d);
                sigma_BG_inv = inv(sigma_BG(1:d,1:d,k));
                PXGY_x_BG = PXGY_x_BG + term1_BG * exp(-0.5 * sum((x_minus_mu_BG * sigma
```

```
            end

            PX_x = PXGY_x_FG * PY_FG + PXGY_x_BG * PY_BG;

            PYGX_FG_x = (PXGY_x_FG * PY_FG) ./ PX_x;
            PYGX_BG_x = (PXGY_x_BG * PY_BG) ./ PX_x;

            itr = 0;
            for i = 1:img_height
                for j = 1:img_width
                    itr = itr + 1;
                    if (PYGX_FG_x(itr,1) > PYGX_BG_x(itr,1))
                        seg_mask_res(i,j) = 1;
                    end
                end
            end

            % compute prob of error for the Mixture models
            prob_error(top_itr, idx) = compute_prob_error(seg_mask_gt, seg_mask_res, PY_
            disp("Prob. Error (d = "+d+"): "+prob_error(top_itr, idx));
        end

        % plot the prob error VS dimension plot here
        figure;
        plot(dim_list, prob_error(top_itr,:), 'color', 'b', 'LineWidth', 2);

        ax.FontSize = 25;
        xlabel("dimension");
        ylabel("Prob. Error [P(error)]");
        title("Probability of Error vs Dimension: Classifier-(FG-"+itr_FG+"/"+max_itr+"
        grid on;
        saveas(gcf, "../plots/Qa/C_FG_"+itr_FG+"_BG_"+itr_BG+"_prob_err_plot.png");
        close;
    end
end

for i = 1:max_itr
    figure;
    for j = 1:max_itr
        plot(dim_list, prob_error((i-1)*5+j,:), 'LineWidth', 2);
        hold on;
    end
    ax.FontSize = 25;
    xlabel("dimension");
    ylabel("Prob. Error [P(error)]");
```

```
    title("Probability of Error vs Dimension: Classifier-(FG-"+i+"/"+max_itr+" , BG-1-TO
    legend("(FG,BG)-("+i+",1)", "(FG,BG)-("+i+",2)", "(FG,BG)-("+i+",3)", "(FG,BG)-("+i+
    grid on;
    saveas(gcf, "../plots/Qa/C_FG_"+i+"_BG_1to5_prob_err_plot.png");
    close;
end
```

2.2 Matlab Code - Function 'EM_algorithm.m'

```
function [mu, sigma, pi_class] = EM_algorithm(TS_DCT, C, d)
    num_sample = size(TS_DCT, 1);

    % randomly initialize the EM-algorithm parameters
    mu = rand(C, d);
    pi_class = rand(C, 1);
    sigma = zeros(d, d, C);
    for j = 1:C
        sigma(:, :, j) = diag(rand(d, 1));
    end

    % start the EM algorithm iteration
    epsilon = 1e-5;
    stop_thresh = 1e-3;

    max_itr = 20;
    itr = 0;
    prev_logL = 1;
    converge = 0;

    % estimating FG class parameters
    while converge == 0
        itr = itr + 1;
        disp(itr+"/"+max_itr);

        h_class = compute_H_FG_BG(TS_DCT, mu, sigma, pi_class, C, d);

        % sum for all training samples and all components dimensions
        sum_h_class = sum(h_class, 'all');
        assert(round(sum_h_class) == num_sample);

        % sum only for all the training samples but not along the components
        % dimension
        sum_h_class_j = transpose(sum(h_class, 1));   % C x 1

        % update pi values
```

```
        pi_class = sum_h_class_j / sum_h_class;   % C x 1

        % update mu values
        mu = (transpose(h_class) * TS_DCT) ./ sum_h_class_j;   % C x d

        % update sigma values
        for j = 1:C
            x_minus_mu = TS_DCT - mu(j, :);
            x_minus_mu_sq = x_minus_mu.^2;
            temp1 = (transpose(h_class(:, j)) * x_minus_mu_sq) / sum_h_class_j(j,1);
            temp1 = max(temp1, epsilon);
            sigma(:, :, j) = diag(temp1);
        end

        logL = compute_logL(TS_DCT, mu, sigma, pi_class, h_class, C, d);

        if (abs((logL - prev_logL)/prev_logL) < stop_thresh) || (itr > max_itr)
            converge = 1;
        end

        prev_logL = logL;
    end
end
```

2.3 Matlab Code - Function 'compute_H_FG_BG.m'

```
function [H] = compute_H_FG_BG(TS_DCT, mu, sigma, pi_class, C, d)
    num_sample = size(TS_DCT, 1);
    H = zeros(num_sample, C);

    for i = 1:num_sample
        for j = 1:C
            H(i, j) = compute_gaussian(TS_DCT(i, :), mu(j, :), sigma(:,:,j), d) * pi_cla
        end
        sum_den = sum(H(i,:), 'all');
        for j = 1:C
            H(i, j) = H(i, j) / sum_den;
        end
    end
end
```

2.4 Matlab Code - Function 'compute_gaussian.m'

```
function [result] = compute_gaussian(x, mu, sigma, d)
    x = reshape(x, [d,1]);
```

```
    mu = reshape(mu, [d,1]);
    sigma = reshape(sigma, [d,d]);

    det_sigma = det(sigma);

%     if (det_sigma == 0)
%         diag_sigma = diag(sigma);
%         diag_sigma = max(diag_sigma, 1e-5);
%         sigma = diag(diag_sigma);
%         det_sigma = det(sigma);
%     end

    sigma_inv = inv(sigma);
    x_minus_mu = x - mu;

    result = (1 / sqrt(power(2*pi, d) * det_sigma)) * exp(-0.5 * transpose(x_minus_mu) *
end
```

2.5 Matlab Code - Function 'compute_logL.m'

```
function [logL] = compute_logL(TS_DCT, mu, sigma, pi_class, h_class, C, d)
    num_sample = size(TS_DCT, 1);

    logL = 0;
    for i = 1:num_sample
        for j = 1:C
            PXGZ_xi_ej = compute_gaussian(TS_DCT(i,:), mu(j,:), sigma(:,:,j), d);
            logL = logL + h_class(i,j) * log(PXGZ_xi_ej * pi_class(j,1));
        end
    end
end
```

2.6 Matlab Code - Function 'compute_prob_error.m'

```
function [prob_error] = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG, PY_BG, meth
    [h, w] = size(seg_mask_gt);

    if (method == 1)
        % Method-1 of computing probability of error
        num_FG_pixels = sum(seg_mask_gt, 'all');
        num_BG_pixels = h * w - num_FG_pixels;

        count_FG_pixels = 0;
        count_BG_pixels = 0;
        count_FG_error = 0;
```

```
        count_BG_error = 0;

        for i = 1:h
            for j = 1:w
                if (seg_mask_gt(i,j) == 1)
                    count_FG_pixels = count_FG_pixels + 1;
                    if (seg_mask_res(i,j) == 0)
                        count_FG_error = count_FG_error + 1;
                    end
                elseif (seg_mask_gt(i,j) == 0)
                    count_BG_pixels = count_BG_pixels + 1;
                    if (seg_mask_res(i,j) == 1)
                        count_BG_error = count_BG_error + 1;
                    end
                end
            end
        end

        assert(num_FG_pixels == count_FG_pixels);
        assert(num_BG_pixels == count_BG_pixels);

        prob_error = (count_FG_error / count_FG_pixels) * PY_FG + (count_BG_error / coun
    elseif (method == 2)
        % Method-2 of computing probability of error
        num_error_pixels = sum(abs(seg_mask_gt - seg_mask_res), 'all');
        num_pixels = h * w;
        prob_error = num_error_pixels / num_pixels;
    end
end
```

2.7 Matlab Code - Function 'img_padding.m'

```
function [img_pad] = img_padding(img)
    [h, w] = size(img);
%     img_pad = zeros(h + 7, w + 7);
%     img_pad(4:h+3, 4:w+3) = img;
%
%     img_pad(1:3, 4:w+3) = img(1:3, :);
%     img_pad(h+4:end, 4:w+3) = img(h-3:h, :);
%     img_pad(4:h+3, 1:3) = img(:, 1:3);
%     img_pad(4:h+3, w+2:end) = img(:, w-7:w-2);
%     img_pad(1:3, 1:3) = img(1:3, 1:3);

    img_pad = padarray(img, [3 3], 'replicate', 'pre');
    img_pad = padarray(img_pad, [4 4], 'replicate', 'post');
```

```
%     img_pad = img(:,:);
%     img_pad(end+1:end+7,:) = img(end-7:end-1,:);
%     img_pad(1:end-7,end+1:end+7) = img(:,end-7:end-1);
%     img_pad(end-7:end,end-7:end) = img(end-7:end,end-7:end);
end
```

2.8 Matlab Code - Function 'main_Qb.m'

```
clc; clear;
close all;

% load the original image and ground-truth segmentation mask
img = im2double(imread("../data/cheetah.bmp"));
img = img(:, 1:end-2);
seg_mask_gt = im2double(imread('../data/cheetah_mask.bmp'));
seg_mask_gt = seg_mask_gt(:, 1:end-2);

% load the zigzag pattern file
zigzag_pat = importdata("../data/zigzag_pattern.txt");
zigzag_pat_lin = zigzag_pat(:) + 1;   % adding 1 for converting to matlab indexes

[img_height, img_width] = size(img);

% pad test image with 7 layers to the right and bottom
img_pad = img_padding(img);

% compute the dct of the padded image once and reuse it for all test
dct_dim = 64;                            % feature dimension
dct_mat = zeros(img_height * img_width, dct_dim);
itr = 0;

for i = 1:img_height
    for j = 1:img_width
        itr = itr + 1;
        img_block_dct = dct2(img_pad(i:i+7, j:j+7));
        dct_mat(itr, zigzag_pat_lin) = img_block_dct(:);
    end
end

disp("Image DCT Computed !!!");

% load the training sample DCT matrix
TS_DCT = load("../data/TrainingSamplesDCT_8_new.mat");
TS_DCT_FG = TS_DCT.TrainsampleDCT_FG;
```

```
TS_DCT_BG = TS_DCT.TrainsampleDCT_BG;

% compute the prior class probabilities
num_sample_FG = size(TS_DCT_FG, 1);
num_sample_BG = size(TS_DCT_BG, 1);
num_sample = num_sample_FG + num_sample_BG;

PY_FG = num_sample_FG / num_sample;
PY_BG = num_sample_BG / num_sample;

d_EM = 64;     % feature size/dimension
C_list = [1,2,4,8,16,32];
C_list_len = size(C_list, 2);
dim_list = [1,2,4,8,16,24,32,40,48,56,64];
num_dim = size(dim_list, 2);

prob_error = zeros(C_list_len, num_dim);

for C_itr = 1:C_list_len

    C = C_list(1, C_itr);
    [mu_FG, sigma_FG, pi_FG] = EM_algorithm(TS_DCT_FG, C, d_EM);
    [mu_BG, sigma_BG, pi_BG] = EM_algorithm(TS_DCT_BG, C, d_EM);

    disp("EM - Parameters Estimation Complete - "+C_itr+"/"+C_list_len);

    for D_itr = 1:num_dim
        d = dim_list(1, D_itr);
        seg_mask_res = zeros(size(img));

        % compute gaussian formula for speed
        num_test_samples = size(dct_mat, 1);
        PXGY_x_FG = zeros(num_test_samples, 1);
        PXGY_x_BG = zeros(num_test_samples, 1);

        for k = 1:C
            term1_FG = 1 / sqrt(power(2 * pi, d) * det(sigma_FG(1:d,1:d,k)));
            x_minus_mu_FG = dct_mat(:,1:d) - mu_FG(k,1:d);
            sigma_FG_inv = inv(sigma_FG(1:d,1:d,k));
            PXGY_x_FG = PXGY_x_FG + term1_FG * exp(-0.5 * sum((x_minus_mu_FG * sigma_FG_

            term1_BG = 1 / sqrt(power(2 * pi, d) * det(sigma_BG(1:d,1:d,k)));
            x_minus_mu_BG = dct_mat(:,1:d) - mu_BG(k,1:d);
            sigma_BG_inv = inv(sigma_BG(1:d,1:d,k));
            PXGY_x_BG = PXGY_x_BG + term1_BG * exp(-0.5 * sum((x_minus_mu_BG * sigma_BG_
```

```
        end

        PX_x = PXGY_x_FG * PY_FG + PXGY_x_BG * PY_BG;

        PYGX_FG_x = (PXGY_x_FG * PY_FG) ./ PX_x;
        PYGX_BG_x = (PXGY_x_BG * PY_BG) ./ PX_x;

        itr = 0;
        for i = 1:img_height
            for j = 1:img_width
                itr = itr + 1;
                if (PYGX_FG_x(itr,1) > PYGX_BG_x(itr,1))
                    seg_mask_res(i,j) = 1;
                end
            end
        end

        % compute prob of error for the Mixture models
        prob_error(C_itr, D_itr) = compute_prob_error(seg_mask_gt, seg_mask_res, PY_FG,
        disp("Prob. Error (d = "+d+"): "+prob_error(1, D_itr));
    end

    % plot the prob error VS dimension plot here
    figure;
    plot(dim_list, prob_error(C_itr,:), 'color', 'b', 'LineWidth', 2);

    ax.FontSize = 25;
    xlabel("dimension");
    ylabel("Prob. Error [P(error)]");
    title("Probability of Error vs Dimension: Classifier - (C = "+C+")");
    grid on;
    saveas(gcf, "../plots/Qb/C_C_"+C+"_prob_err_plot.png");
    close;
end

figure;
for C_itr = 1:C_list_len
    plot(dim_list, prob_error(C_itr,:), 'LineWidth', 2);
    hold on;
end

ax.FontSize = 25;
xlabel("dimension");
ylabel("Prob. Error [P(error)]");
title("Probability of Error vs Dimension: Classifier - (C = [1,2,4,8,16,32])");
```

```
legend("C=1","C=2","C=4","C=8","C=16","C=32");
grid on;
saveas(gcf, "../plots/Qb/C_C_all_prob_err_plot.png");
close;
```