ECE 271A - Homework assignment

Taruj Goyal

October 31, 2019

1 Problem Formulation

- **Goal:** Segment the image into the cheetah (foreground) and grass (background).
- **Problem Setup as Pattern Recognition:**
  - **Observation Space:** Images as a collection of 8×8 blocks.
  - **Feature Computation:**
    - For each block compute the 2D Discrete Cosine Transform.
    - Reorder the 8×8 frequency decomposition into Zig-Zag pattern.
    - Resize the 8×8 array into 64D feature vector and use it as a feature vector.

2 Results

- **(a) Prior Probability**: $\Pr(cheetah)$ & $\Pr(grass)$
  - Based on the results of the second problem the we obtain the maximum likelihood estimate for the prior probabilities as $\pi_j^* = c_j/n$.
  - c_j is the number of occurrences of the class j.
  - n is the total number of occurrences in the dataset.
  - This can be computed using the dimensions TrainingSamplesCD_8.mat
  - The number of training examples for the Cheetah and Grass are 250 and 1053, respectively.
  - $\Pr(cheetah) = \frac{250}{250+1053}$
  - $\Pr(grass) = \frac{1053}{250+1053}$
  - **Based on these dimensions** $\Pr(cheetah)$ **= 0.1919 &** $\Pr(grass)$ **= 0.8081.**
  - The priors with the maximum likelihood estimates are the same as we obtained last week. And this shows that the Maximum Likelihood Estimate is effectively what we chose intuitively based on the relative presence of grass and cheetah in the training data.

- Hence, the relative number of occurrences of a particular class is a measure for the prior for a class.

- **(b) Class Conditional Probability (Marginal)**: $\Pr(x \mid cheetah)$ & $\Pr(x \mid grass)$
  - The maximum likelihood estimates for the parameters of the class conditional densities under the Gaussian assumption are the following:
  - $$\mu = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$
  - $$\boldsymbol{\Sigma} = \frac{1}{n} \sum_{i} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$
  - For obtaining each individual distribution for each feature we can use the diagonal elements of the covariance matrix and the corresponding value in the mean vector.
  - On plotting the 64 different features we get the following conditional distributions.
  - $$P_{X|Y}(x|i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left\{-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right\}$$
  - **Reducing Feature set based on inspection:**
  - **We choose those marginal distributions that have a minimum overlap and different spreads.**
  - The best features visually on inspection are 1,13,19,26,29,32,33,and 40.
  - The worst features visually on inspection are 3,4,5,59,60,62,63,and 64.

- **(c) - Decision Rule**
  - We use the following decision rules for multivariate gaussians without equal covariance matrices.
  - $$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + w_{i0}$$
  - $$\mathbf{W}_i = -\frac{1}{2} \boldsymbol{\Sigma}_i^{-1}$$
  - $$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i$$
  - $$w_{i0} = -\frac{1}{2} \boldsymbol{\mu}_i^t \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(i)$$
  - We classify it using the 64-dimensional Gaussians and the 8-dimensional Gaussians associated with the best 8 features.

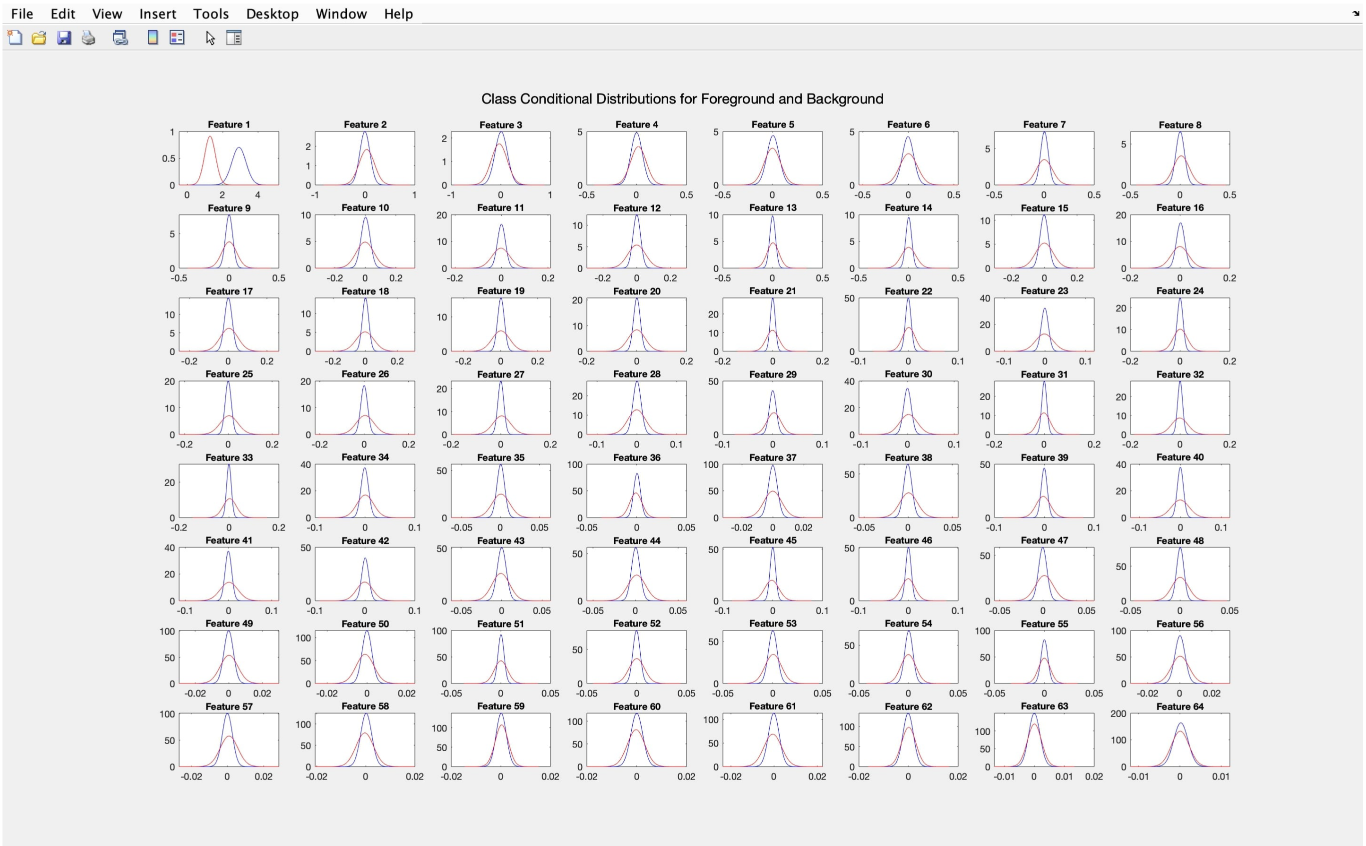

Figure 1: Class Conditional Densities: $\Pr(x \mid y)$

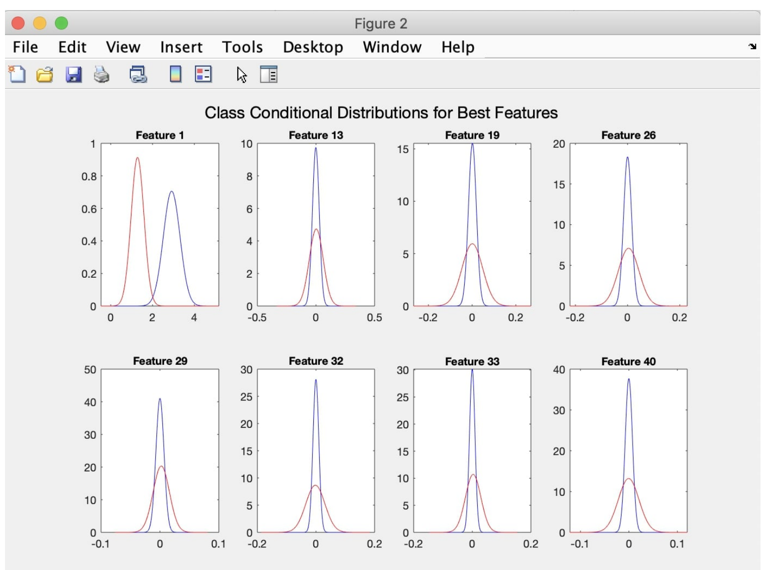

Figure 2: Best Eight Features

- Basically for each input pixel, we compute the 64-dimensional feature vector x and for the 8-dimensional gaussian we select the corresponding 8 elements of the

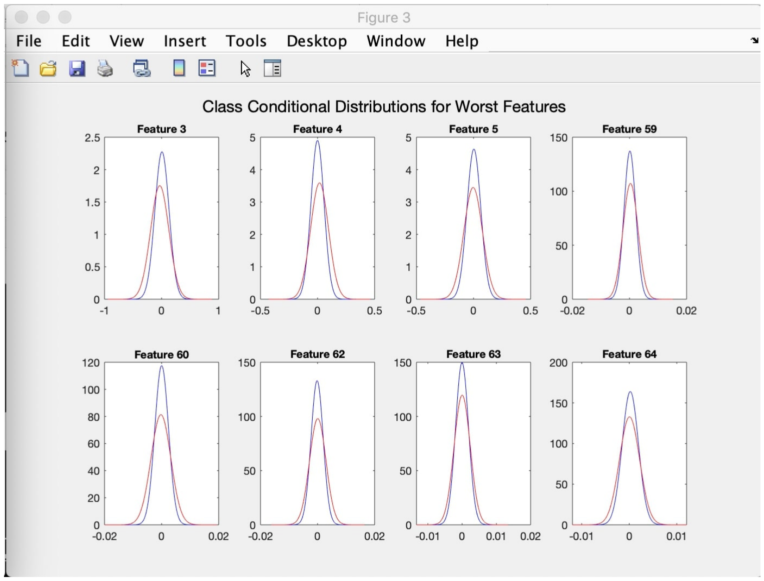

Figure 3: Worst Eight Features

64 dimensional feature vector. And plug it into the the decision rule above and then chose the class with the higher value of $g_i(x)$.

- Based on this we get the classification output for that pixel.

- **Computing the Probability of Error**

  - By comparing the pixel-wise mismatches between the ground truth and the generated segmentation map the error is computed.
  - Average probability of error is $R^* = \int P_{X,Y}(x, y \neq g^*(x))dx$.
  - Let cheetah = 1 and grass = 0,
  - The above equation boils down to $P_{error} = P(y \neq g(x))$
  - $P_{error} = P(y = 1)P(g(x) = 0|y = 1) + P(y = 0)P(g(x) = 1|y = 0)$.
  - This is basically the number of pixels mismatching between the predicted output and ground truth divided by the total number of pixels.
  - **The probability of error is 0.137588 for the 64-dimensional Gaussians**
  - **The probability of error is 0.080189 for the e 8-dimensional Gaussians associated with the best 8 features**
  - On using lesser number of features we get a better set of features and the remaining features maybe corrupted by noise. These few features are those which discriminate amongst the classes more. The one with 64 features have quite a few features which are similar for both classes and may lead to poor results henceforth. These are not as pronounced as the best features.

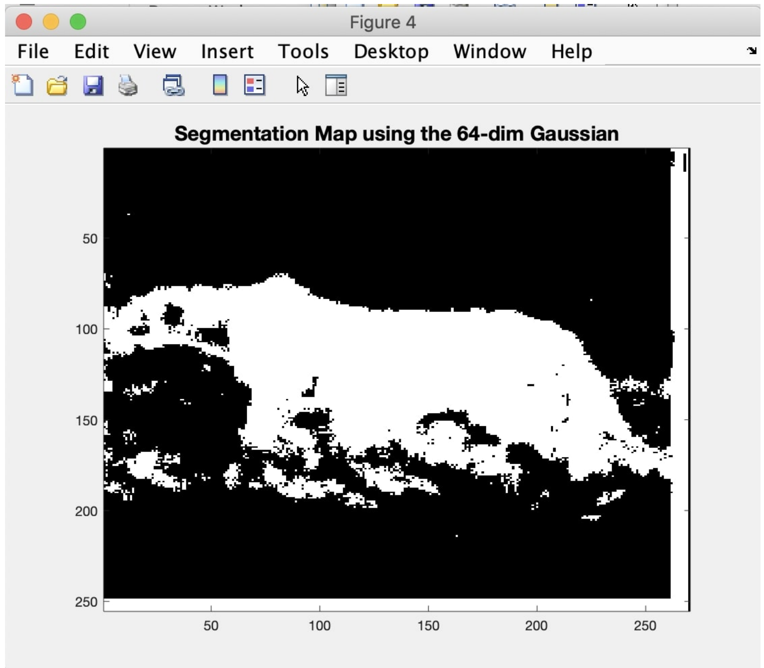

Figure 4: Segmentation map for 64-dimensional Gaussians

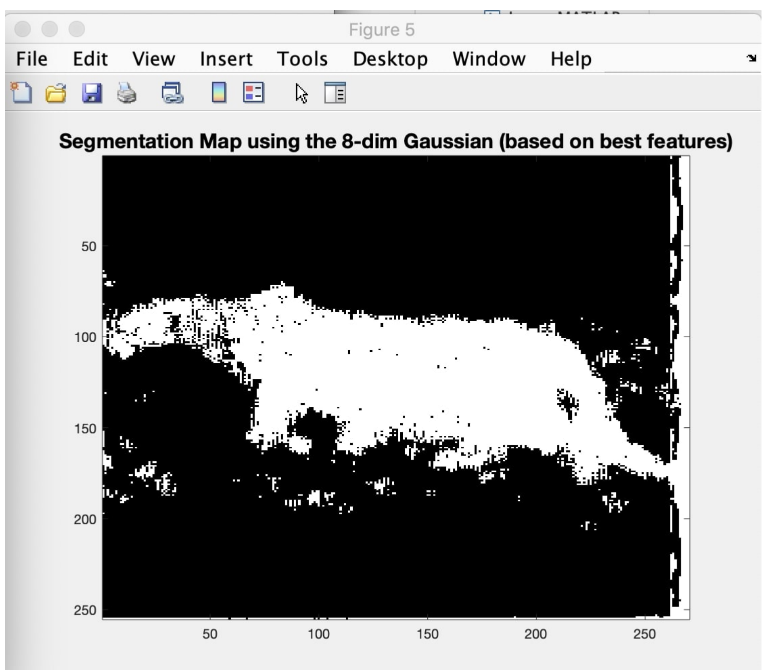

Figure 5: Segmentation map for 8 best features

3 Appendix

Listing 1: MATLAB Solution Script

```
%% Clear workspace and close windows

    clc
    clear
    close all

%% Load training dataset

    DCT = load('TrainingSamplesDCT_8.mat');
    Train_DCT_FG = DCT.TrainsampleDCT_FG;
    Train_DCT_BG = DCT.TrainsampleDCT_BG;

%% The class priors for FG and BG are computed
%  Section A

    % Assuming prior distribution is (# of samples of A)/total training ...
       samples
    TotalNumberOfTraining_samples = size(Train_DCT_FG,1) + ...
       size(Train_DCT_BG,1);
    FG_prior = size(Train_DCT_FG,1)/TotalNumberOfTraining_samples;
    BG_prior = size(Train_DCT_BG,1)/TotalNumberOfTraining_samples;

%% Training data used for computing and plotting index histograms.

    Mean_BG = ...
       Train_DCT_BG'*(ones(size(Train_DCT_BG,1),1))/size(Train_DCT_BG,1);
    Mean_FG = ...
       Train_DCT_FG'*(ones(size(Train_DCT_FG,1),1))/size(Train_DCT_FG,1);

    Var_BG = (Train_DCT_BG'*Train_DCT_BG)/size(Train_DCT_BG,1) - ...
       Mean_BG*Mean_BG';
    Var_FG = (Train_DCT_FG'*Train_DCT_FG)/size(Train_DCT_FG,1) - ...
       Mean_FG*Mean_FG';

    figure(1)
    fontSize = 10;

    for i = 1:64

        start_range = ...
           min(Mean_BG(i)-4*sqrt(Var_BG(i,i)),Mean_FG(i)-4*sqrt(Var_FG(i,i)));
        end_range = ...
           max(Mean_FG(i)+4*sqrt(Var_FG(i,i)),Mean_BG(i)+4*sqrt(Var_BG(i,i)));
        step_size = 0.01*min(sqrt(Var_FG(i,i)),sqrt(Var_BG(i,i)));

        x = start_range:step_size:end_range;

        BG_marginal = ...
           (1/sqrt(2*pi*Var_BG(i,i)))*exp(-((x-Mean_BG(i)).^2/Var_BG(i,i)));
        FG_marginal = ...
           (1/sqrt(2*pi*Var_FG(i,i)))*exp(-((x-Mean_FG(i)).^2/Var_FG(i,i)));
        subplot(8,8,i);
```

```
        plot(x,BG_marginal,'b',x,FG_marginal,'r')

        title(sprintf('Feature %d',i), 'FontSize', fontSize)
    end
    sgtitle('Class Conditional Distributions for Foreground and ...
        Background','FontSize', 1.5*fontSize)

%% For each block in the image cheetah.bmp, compute the feature X and ...
    state variable Y.

    block_size = 8;

    % Read the ZigZag pattern and convert to array and index from 1
        ZigZagPattern = table2array(readtable('Zig-Zag Pattern.txt'))+1;

    % Read image and convert to double
        image = im2double(imread('cheetah.bmp'));

    % Reading image dimensions
        [height,width] = size(image);
        [h_8,w_8] = deal(8*(ceil(height/8)+1),8*(ceil(width/8)+1));

    % Zeropad the image and convert to 8x8
        zeropad = zeros(h_8,w_8);
        zeropad(1:height,1:width) = image;

    % Create a blank array X
        X = zeros(height,width);
        Y = zeros(height,width);
        dctZigZag = zeros(1,block_size*block_size);

    %%  Computation of metrics for the best 8 features
        precision_BG = inv(Var_BG);
        precision_FG = inv(Var_FG);

        best_8 = [1,13,19,26,29,32,33,40];
        worst_8 = [3,4,5,59,60,62,63,64];


        Mean_BG_8_best = Mean_BG(best_8);
        Mean_FG_8_best = Mean_FG(best_8);

        Var_BG_8_best = Var_BG(best_8,best_8);
        Var_FG_8_best = Var_FG(best_8,best_8);

        precision_BG_8_best = inv(Var_BG_8_best);
        precision_FG_8_best = inv(Var_FG_8_best);


    %%
        figure(2)
        for i = 1:8
            idx = best_8(i);
```

```
            start_range = ...
                min(Mean_BG(idx)-4*sqrt(Var_BG(idx,idx)),Mean_FG(idx)-4*sqrt(Var_FG(idx,
            end_range = ...
                max(Mean_FG(idx)+4*sqrt(Var_FG(idx,idx)),Mean_BG(idx)+4*sqrt(Var_BG(idx,
            step_size = ...
                0.01*min(sqrt(Var_FG(idx,idx)),sqrt(Var_BG(idx,idx)));

            x = start_range:step_size:end_range;

            BG_marginal = ...
                (1/sqrt(2*pi*Var_BG(idx,idx)))*exp(-((x-Mean_BG(idx)).^2/Var_BG(idx,idx)
            FG_marginal = ...
                (1/sqrt(2*pi*Var_FG(idx,idx)))*exp(-((x-Mean_FG(idx)).^2/Var_FG(idx,idx)
            subplot(2,4,i);
            plot(x,BG_marginal,'b',x,FG_marginal,'r')

            title(sprintf('Feature %d',idx), 'FontSize', fontSize)
        end
        sgtitle('Class Conditional Distributions for Best ...
            Features','FontSize', 1.5*fontSize)

        figure(3)

        for i = 1:8
            idx = worst_8(i);
            start_range = ...
                min(Mean_BG(idx)-4*sqrt(Var_BG(idx,idx)),Mean_FG(idx)-4*sqrt(Var_FG(idx,
            end_range = ...
                max(Mean_FG(idx)+4*sqrt(Var_FG(idx,idx)),Mean_BG(idx)+4*sqrt(Var_BG(idx,
            step_size = ...
                0.01*min(sqrt(Var_FG(idx,idx)),sqrt(Var_BG(idx,idx)));

            x = start_range:step_size:end_range;

            BG_marginal = ...
                (1/sqrt(2*pi*Var_BG(idx,idx)))*exp(-((x-Mean_BG(idx)).^2/Var_BG(idx,idx)
            FG_marginal = ...
                (1/sqrt(2*pi*Var_FG(idx,idx)))*exp(-((x-Mean_FG(idx)).^2/Var_FG(idx,idx)
            subplot(2,4,i);
            plot(x,BG_marginal,'b',x,FG_marginal,'r')

            title(sprintf('Feature %d',idx), 'FontSize', fontSize)
        end
        sgtitle('Class Conditional Distributions for Worst ...
            Features','FontSize', 1.5*fontSize)

    %% Generating the output image for the two features

    % Iterating through the image in 8x8 blocks through a sliding window
    for h = 1:height
        for w = 1:width

            dctBlock = dct2(zeropad(h:h+block_size-1,w:w+block_size-1));

```

```
            % Rearranging the DCT Components in ZigZag Patterned Vector
            for i = 1:block_size
                for j = 1:block_size
                    dctZigZag(ZigZagPattern(i,j)) = dctBlock(i,j);
                end
            end

            g_grass = ...
               decision_bound(dctZigZag',Var_BG,Mean_BG,BG_prior,precision_BG);
            g_cheetah = ...
               decision_bound(dctZigZag',Var_FG,Mean_FG,FG_prior,precision_FG);
            % Computing the feature for the block
            X(h,w) = g_grass > g_cheetah;

            g_grass_8 = ...
               decision_bound(dctZigZag(best_8)',Var_BG_8_best,Mean_BG_8_best,BG_prior,
            g_cheetah_8 = ...
               decision_bound(dctZigZag(best_8)',Var_FG_8_best,Mean_FG_8_best,FG_prior,

            Y(h,w) = g_grass_8 > g_cheetah_8;
        end
    end

    %% Displaying the output image for both feature sets
    A = X(1:height,1:width);
    B = Y(1:height,1:width);

    figure(4)
    imagesc(A);
    colormap(gray(255));
    imwrite(A, 'result.bmp');
    title('Segmentation Map using the 64-dim Gaussian', 'FontSize', ...
       1.5*fontSize);

    figure(5)
    imagesc(B);
    colormap(gray(255));
    imwrite(B, 'result_8best.bmp');
    title('Segmentation Map using the 8-dim Gaussian (based on best ...
       features)', 'FontSize', 1.5*fontSize);

%% Compare the ground truth in image cheetah_mask.bmp and compute the ...
   probability of error

    % Read the ground truth image
        ground_truth = im2double(imread('cheetah_mask.bmp'));

        sprintf('Error for the 64-dim gaussian ...
           %f',error_computation(ground_truth,A,FG_prior,BG_prior))
        sprintf('Error for the 8-dim gaussian ...
           %f',error_computation(ground_truth,B,FG_prior,BG_prior))

%% UTILITY FUNCTIONS

```

```
    function [g_x] = decision_bound(x,Var,Mean,prior,precision)

        w_i_0 = log(det(Var))-2*log(prior)+(Mean'*precision*Mean);
        w_i = -2*precision*Mean;
        g_x = x'*precision*x + w_i'*x + w_i_0;
    end

    function [probability_error] = ...
       error_computation(ground_truth,prediction,FG_prior,BG_prior)

        % Probability of error for Cheetah pixels misclassified as Grass
            probability_error_cheetah = sum(ground_truth & ¬...
                prediction,'all')/sum(ground_truth,'all');
        % Probability of error for Grass pixels misclassified as Cheetah
            probability_error_grass = sum(¬ground_truth & ...
                prediction,'all')/sum(¬ground_truth,'all');
        % Computation of probability of error
            probability_error = (FG_prior*probability_error_cheetah) + ...
                (BG_prior*probability_error_grass);
    end
```