ECE 271A - Statistical Learning I - Assignment 1

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

October 12, 2021

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

Figure 1: Original Cheetah Image

Figure 2: GT Segmentation Mask

Question 5a. Using the training data in TrainingSamplesDCT_8.mat, what are reasonable estimates for the prior probabilities?

Solution: The training sample DCT contains two DCT components for foreground and background classes. These training vectors have been extracted from a similar image as the cheetah image. The number of foreground and background training vectors provide an indication of the number of 8x8 patches in the training image consisting of foreground and background classes resp. Therefore, the prior probabilities can be reasonably estimated in the following way.

Number of 64-D vectors in foreground class (N_{FG}) = 250

Number of 64-D vectors in background class (N_{BG}) = 1053
Total number of 64-D training vectors (N) = 250+1053 = 1303

$$P_Y(\text{cheetah}) = N_{FG}/N = 0.1919 \tag{1}$$

$$P_Y(\text{grass}) = N_{BG}/N = 0.8081 \tag{2}$$

Question 5b. Using the training data in TrainingSamplesDCT_8.mat, compute and plot the index histograms $P_{X|Y}(x|\text{cheetah})$ and $P_{X|Y}(x|\text{grass})$.

Solution: Each row in the training sample BG (1053x64) and FG (250x64) represents a training vector for the corresponding class. Therefore, the 1-D feature is first extracted from each row using the index of the second largest DCT coefficient. Each feature X lies between 1 and 64. The histogram plot (with probability normalization) of these features gives the class-conditional probabilities as shown below.

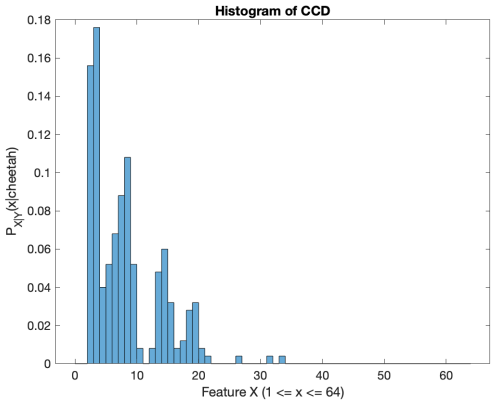

Figure 3: CCD for Y=cheetah

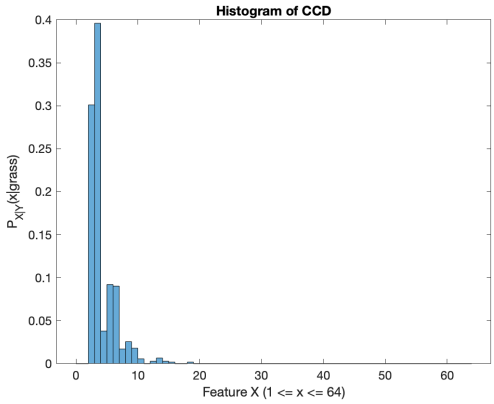

Figure 4: CCD for Y=grass

Question 5c. For each block in the image cheetah.bmp, compute the feature X (index of the DCT coefficient with 2nd greatest energy). Compute the state variable Y using the minimum probability of error rule based on the probabilities obtained in a) and b). Store the state in an array A. Using the commands imagesc and colormap(gray(255)) create a picture of that array.

Solution: From Q5a, we know the prior probabilities ($P_Y(\text{cheetah})$ and $P_Y(\text{grass})$), and from Q5b we know the class-conditional probabilities ($P_{X|Y}(x|\text{cheetah})$ and $P_{X|Y}(x|\text{grass})$).

$$P_{X,Y}(x, \text{cheetah}) = P_{X|Y}(x|\text{cheetah})P_Y(\text{cheetah}) \tag{3}$$

$$P_{X,Y}(x, \text{grass}) = P_{X|Y}(x|\text{grass})P_Y(\text{grass}) \tag{4}$$

$$P_X(x) = P_{X,Y}(x, \text{grass}) + P_{X,Y}(x, \text{cheetah}) \tag{5}$$

$$P_{Y|X}(\text{cheetah}|x) = \frac{P_{X|Y}(x|\text{cheetah})P_Y(\text{cheetah})}{P_X(x)} \tag{6}$$

$$P_{Y|X}(\text{grass}|x) = \frac{P_{X|Y}(x|\text{grass})P_Y(\text{grass})}{P_X(x)} \tag{7}$$

Optimal decision function of 0-1 loss is the MAP rule

$$g^*(x) = \text{argmax}P_{Y|X}(i|x) \tag{8}$$

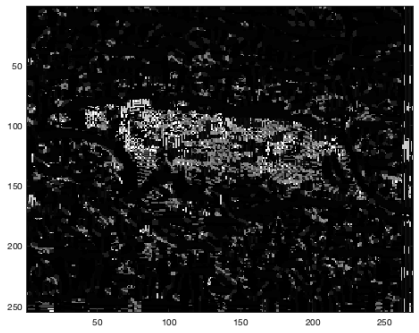

Figure 5: Foreground Mask ($P_{Y|X}(cheetah|x)$)

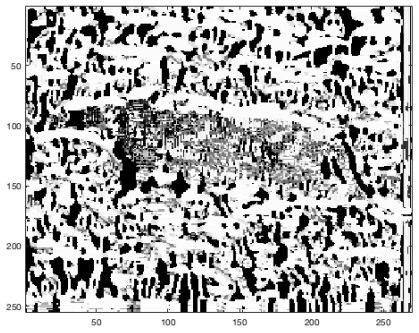

Figure 6: Background Mask ($P_{Y|X}(grass|x)$)

Figure 7: GT Segmentation Mask

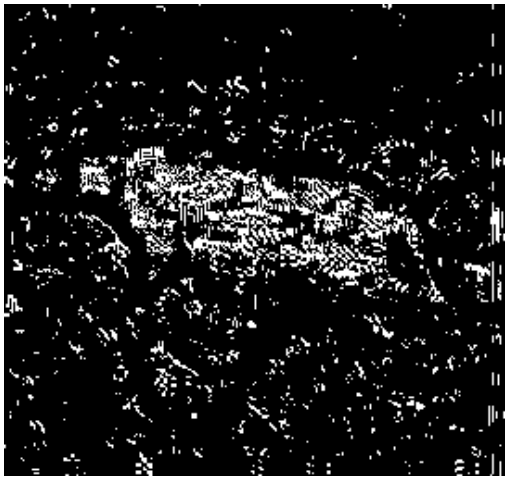

Figure 8: Segmentation Mask Result

Question 5d. The array A contains a mask that indicates which blocks contain grass and which contain the cheetah. Compare it with the ground truth provided in image cheetah mask.bmp (shown below on the right) and compute the probability of error of your algorithm.

Solution: For estimating the probability of error, the ground-truth segmentation mask is compared with the resulting segmentation mask.
Let total number of pixels in the image be N, number of correctly predicted pixels in the

segmentation mask be denoted as N_{corr}, and number of incorrectly predicted pixels in the segmentation mask be denoted as N_{incorr}. Then error percentage is -

$$\text{error} = N_{incorr}/N = 0.185 \tag{9}$$

Probability of error of the decision rule $g^*(x)$ is given by the minimum risk associated with the optimal decision.

$$P(\text{error}) = R^* = \sum_{i=1}^{64} P_{Y,X}(y \neq g^*(x), x) = \sum_{i=1}^{64} P_{Y|X}(y \neq g^*(x)|x)P_X(x) = \mathbf{0.2135} \tag{10}$$

2 Appendix

Main Code - 'main.m'

```
clc; clear;
img = im2double(imread("../data/cheetah.bmp"));
[img_height,img_width] = size(img);
% imwrite(img, '../plots/original_img.png');

% 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

% figure;
% imshow(img);

% load the training sample DCT matrix
train_sample_DCT = load("../data/TrainingSamplesDCT_8.mat");
train_sample_DCT_FG = train_sample_DCT.TrainsampleDCT_FG;
train_sample_DCT_BG = train_sample_DCT.TrainsampleDCT_BG;

% compute the index histograms for FG and BG training samples
FG_feature = find_feature(train_sample_DCT_FG);   % P(x|cheetah)
BG_feature = find_feature(train_sample_DCT_BG);   % P(x|grass)

figure;
h_FG = histogram(FG_feature, 64, 'BinWidth', 1, 'Normalization', 'probability', 'BinEdge
% legend('PX|Y(x|cheetah)');
ax = gca;
ax.FontSize = 12;
xlabel('Feature X (1 <= x <= 64)');
ylabel('P_{X|Y}(x|cheetah)');
```

```
title('Histogram of CCD');
% saveas(gcf, "../plots/FG_hist.png");
PXGY_x_C = h_FG.Values;

figure;
h_BG = histogram(BG_feature, 64, 'BinWidth', 1, 'Normalization', 'probability', 'BinEdge
% legend('PX|Y(x|cheetah)');
ax = gca;
ax.FontSize = 12;
xlabel('Feature X (1 <= x <= 64)');
ylabel('P_{X|Y}(x|grass)');
title('Histogram of CCD');
% saveas(gcf, "../plots/BG_hist.png");
PXGY_x_G = h_BG.Values;

% prior probabilities for cheetah and grass
num_sample_FG = size(train_sample_DCT_FG,1);
num_sample_BG = size(train_sample_DCT_BG,1);
total_samples = num_sample_FG + num_sample_BG;

PY_C = num_sample_FG/total_samples;   % P(Y=cheetah)
PY_G = num_sample_BG/total_samples;   % P(Y=grass)

PXY_x_C = PXGY_x_C*PY_C;   % joint probability of X and Y
PXY_x_G = PXGY_x_G*PY_G;

PX_x = PXY_x_C + PXY_x_G;   % marginalization in Y to obtain P(X)

% classification of each pixel into cheetah and grass
img_dct = zeros(size(img));
block_dct_vec = zeros(1,64);
seg_mask_res = zeros(size(img));

% pad test image with 7 layers to the right and bottom
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);

PYGX_C_x = zeros(size(img));
PYGX_G_x = zeros(size(img));

img_feature_map = zeros(size(img));

for i=1:img_height
```

```
    for j=1:img_width
        img_block_dct = dct2(img_pad(i:i+7,j:j+7));
        block_dct_vec(1,zigzag_pat_lin) = img_block_dct(:);

        block_dct_feature = find_feature(block_dct_vec);
        img_feature_map(i,j) = block_dct_feature;

        PYGX_C_x(i,j) = (PXGY_x_C(block_dct_feature)*PY_C)/PX_x(block_dct_feature);
        PYGX_G_x(i,j) = (PXGY_x_G(block_dct_feature)*PY_G)/PX_x(block_dct_feature);

        if (PYGX_C_x(i,j) >= PYGX_G_x(i,j))
            seg_mask_res(i,j) = 1;
        else
            seg_mask_res(i,j) = 0;
        end
    end
end

figure;
FG_img_prob = imagesc(PYGX_C_x);
colormap(gray(255));
% imwrite(FG_img_prob, "../plots/FG_colormap.tiff");

figure;
BG_img_prob = imagesc(PYGX_G_x);
colormap(gray(255));
% imwrite(BG_img_prob, "../plots/BG_colormap.tiff");

% figure;
% imshow(img_pad);

figure;
imshow(seg_mask_res);
% imwrite(seg_mask_res, "../plots/seg_mask_res.png");

% load the ground-truth segmentation mask
seg_mask_gt = im2double(imread('../data/cheetah_mask.bmp'));
% imwrite(seg_mask_gt, '../plots/seg_mask_gt.png');
figure;
imshow(seg_mask_gt);

% estimate probability error using segmentation result and ground-truth
num_pixels = size(seg_mask_gt,1)*size(seg_mask_gt,2);
num_corr_pred = sum(seg_mask_gt == seg_mask_res, 'all');
num_incorr_pred = sum(seg_mask_gt ~= seg_mask_res, 'all');
```

```
error_frac = num_incorr_pred/num_pixels;

% estimate the minimum prob. of error
final_prob_error = 0;
for i=1:64
    num_pixels = sum(img_feature_map(:)==i);  % num pixels with feature=i
    if num_pixels ~= 0
        num_wrong_pred = sum(abs(seg_mask_res(img_feature_map==i) - seg_mask_gt(img_feat
        final_prob_error = final_prob_error + (num_wrong_pred/num_pixels)*PX_x(i);
    end
end
```

Dependency Code - 'find_feature.m'

```
function img_feature = find_feature(img_dct)
    [num_rows,~] = size(img_dct);
    % disp(num_rows);
    img_feature = zeros(num_rows,1);
    img_dct_abs = abs(img_dct);

    % convert 64-D feature to 1-D feature using 2nd largest energy
    for i=1:num_rows
        [~,max_idx] = max(img_dct_abs(i,:));    % 1st largest coefficient
        img_dct_abs(i,max_idx) = -Inf;
        [~,max_idx] = max(img_dct_abs(i,:));    % 2nd largest coefficient

        img_feature(i,1) = max_idx;
    end
end
```