Student Name: Lin-Hsi (Ethen) Tsao
Student PID: A59015911

Statistical Learning 1 (ECE 271A)

HW1

(a) **using the training data in TrainingSamplesDCT 8.mat, what are reasonable estimates for the prior probabilities?**

$$\text{size of total training sample} = \text{size of cheetah training sample} + \text{size of grass training sample}$$

$$P_Y(cheetah) = \frac{\text{size of cheetah training sample}}{\text{size of total training sample}}$$

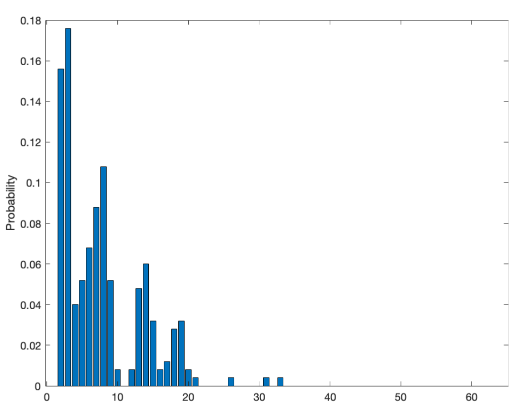
$$P_Y(grass) = \frac{\text{size of grass training sample}}{\text{size of total training sample}}$$

In this case, $P_Y(cheetah) = 0.1919, P_Y(grass) = 0.8081$.

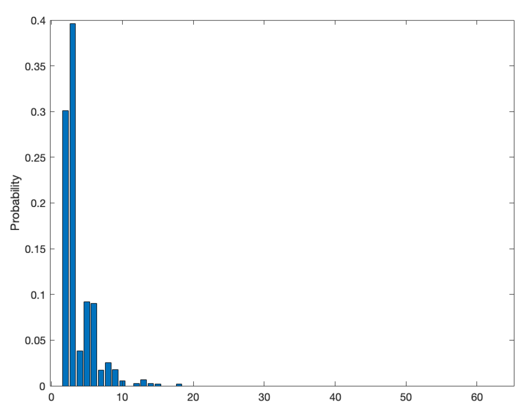
(b) **using the training data in TrainingSamplesDCT 8.mat, compute and plot the index histograms $P_{(X|Y)}(x|cheetah)$ and $P_{(X|Y)}(x|grass)$.**

For this problem, we are required to compute the index histograms $P_{(X|Y)}(x|cheetah)$ and $P_{(X|Y)}(x|grass)$. To get the answer, there are 5 steps to be followed:

1) For each row in $TrainsampleDCT\_FG$, find the index of the second largest value in $TrainsampleDCT\_FG(row)$.
2) Count the frequency of the index and store it in the $freqFG$ array.
3) Let all the element in the $freqFG$ array be in a range of [0, 1] by dividing length($TrainsampleDCT\_FG$).
4) Plot the histogram.
5) Repeat the steps mentioned above by changing $TrainsampleDCT\_FG$ to $TrainsampleDCT\_BG$, $freqFG$ to $freqBG$.



(a) $P_{(X|Y)}(x|cheetah)$



(b) $P_{(X|Y)}(x|grass)$

(c) **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.**

For this problem, 5 steps are to be followed:

1) To normalize the value in the range [0, 1], divide img *cheetah.bmp* by 255.
2) Padding the img *cheetah.bmp* with 0 for all four sides to get a new image size with (263, 278). After performing sliding windows, we could get the matrix with the size of (255, 270), the same as the input img, by padding.
3) For each 8*8 block, we compute DCT, order coefficients with zig-zag scan, and pick position of the second largest magnitude as the feature value.
4) Determine the predicted class in the current block by BDR. Noticed that *index* is in the range [1, 64].

   if $priorFG * freqFG(index) > priorBG * freqBG(index)$, predicted class = 1
   else, predicted class = 0

5) Create a binary mask A with 1's for cheetah (foreground) and 0's for grass (background).

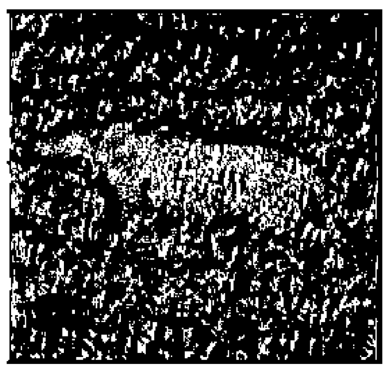


Figure 2: Binary mask A

(d) **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.**

For all the pixel in the binary mask A, we could check whether the predicted label equals to the ground truth label in *cheetah_mask.bmp*. The error rate could be calculated by

$$\text{error rate} = \frac{\text{number of different pixel value}}{\text{number of pixels in the mask}}$$

In this case, error rate = 0.2559.

Code

```matlab
load('TrainingSamplesDCT_8.mat');
total = size(TrainsampleDCT_FG) + size(TrainsampleDCT_BG);
priorFG = length(TrainsampleDCT_FG) / total(1);
priorBG = length(TrainsampleDCT_BG) / total(1);
```

Figure 3: Code for problem (a)

```matlab
load('TrainingSamplesDCT_8.mat');
freqFG = zeros(1, 64);
freqBG = zeros(1, 64);

% pick position of 2nd largest magnitude as the feature value
for row = 1 : length(TrainsampleDCT_FG)
    flatBlock = TrainsampleDCT_FG (row, :);
    [secondLargeNum, secondLargeIdx] = maxk(flatBlock, 2);
    freqFG(1, secondLargeIdx(2)) = freqFG(1, secondLargeIdx(2)) + 1;
end

% compute probability;
freqFG = freqFG / length(TrainsampleDCT_FG);

% plot histogramFG;
bar(linspace(1, 64, 64), freqFG);
xlabel('Pos');
ylabel('Probability')
savefig('histogramFG.fig');

% pick position of 2nd largest magnitude as the feature value
for row = 1 : length(TrainsampleDCT_BG)
    flatBlock = TrainsampleDCT_BG(row, :);
    [secondLargeNum, secondLargeIdx] = maxk(flatBlock, 2);
    freqBG(1, secondLargeIdx(2)) = freqBG(1, secondLargeIdx(2)) + 1;
end

% compute probability;
freqBG = freqBG / length(TrainsampleDCT_BG);

% plot histogramBG;
bar(linspace(1, 64, 64), freqBG);
xlabel('Pos');
ylabel('Probability')
savefig('histogramBG.fig');
```

Figure 4: Code for problem (b)

```matlab
% read img;
img = imread('cheetah.bmp');
img = double(img) / 255;

% padding img with zero;
I = zeros(263, 278);
for row = 5 : 259
    for col = 5 : 274
        I(row, col) = img(row - 4, col - 4);
    end
end

% read pattern;
pattern = readmatrix('Zig-Zag Pattern.txt');

A = zeros(255, 270);

for row = 1 : 255
    for col = 1 : 270
        block = zeros(8, 8);
        % get the blcok;
        for r = row : row + 7
            for c = col : col + 7
                block(r - row + 1, c - col + 1) = I(r, c);
            end
        end

        % compute DCT;
        dct2Block = dct2(block);
        flatBlock = zeros(1, 64);

        % order coefficients with zig-zag scan;
        for r = 1 : 8
            for c = 1 : 8
                flatBlock(1, pattern(r, c) + 1) = dct2Block(r, c);
            end
        end

        % pick position of 2nd largest magnitude as the feature value;
        [secondLargeNum, secondLargeIdx] = maxk(flatBlock, 2);

        % use BDR to find class Y for each block;
        % create a binary mask;
        if priorFG * freqFG(secondLargeIdx(2)) > priorBG * freqBG(secondLargeIdx(2))
            A(row, col) = 1;
        else
            A(row, col) = 0;
        end
    end
end

imshow(uint8(A), [0 1]);
savefig('A.fig');
```

Figure 5: Code for problem (c)

```matlab
mask = imread('cheetah_mask.bmp');
errorCount = 0;
sizes = size(mask);
rows = sizes(1);
cols = sizes(2);

% check whether the predicted label equals to the ground truth label;
for row  = 1 : rows
    for col  = 1 : cols
        if (mask(row, col) / 255 ~= A(row, col))
            errorCount = errorCount + 1;
        end
    end
end


% compute error rate;
errorRate = errorCount / (rows * cols);
disp(errorRate);
```

Figure 6: Code for problem (d)