# Homework Set One
# ECE 271A

Name : Hsiu-Wen Yen

ID : A59010599

a) Prior probability of cheetah can be estimated by dividing total cheetah counts in training sample with total pixel counts. Same for estimating prior probability of grass.

Ans:

P(cheetah) = 0.1919

P(Grass) = 0.8081

```
1  clear; clc;
2  % load trainging sample and image
3  load('TrainingSamplesDCT_8.mat');
4  cheetah = imread('cheetah.bmp');
5  % calculate prior probabilities of cheetah and grass
6  pixel_total_count = size(TrainsampleDCT_FG, 1) + size(TrainsampleDCT_BG, 1);
7  prior_Pcheetah = size(TrainsampleDCT_FG, 1) / pixel_total_count;
8  prior_Pgrass = size(TrainsampleDCT_BG, 1) / pixel_total_count;
```

Command Window

New to MATLAB? See resources for Getting Started.

```
>> prior_Pcheetah
prior_Pgrass

prior_Pcheetah =

    0.1919


prior_Pgrass =

    0.8081

>>
```

b) Choosing index of second large magnitude in DCT coefficient as X variable.

Defining **find_second_large** function to get x variable for each dct coefficient matrix.

```
function x_index = find_second_large(data)
    for index = 1 : size(data)
        arr = abs(data);
        x_value = max(arr(arr<max(arr)));
        x_index = find(arr == x_value);
    end
end
```

Calculate frequency of X variable by traverse all pixels' dct coefficients in training sample.

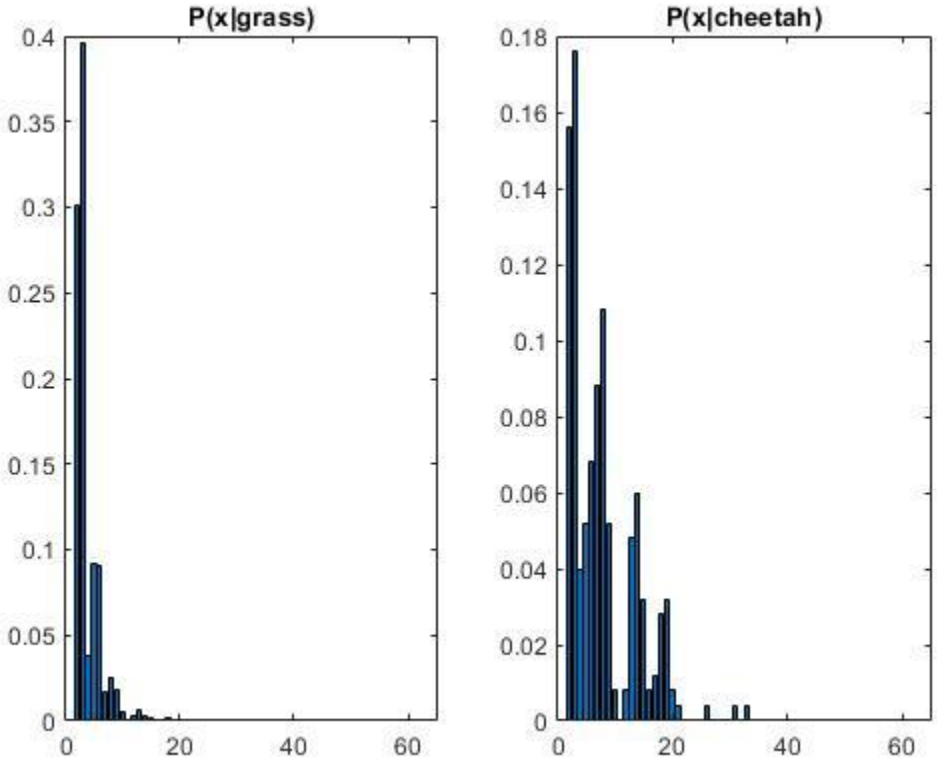
```
% calculate P(x|cheetah) and P(x|grass)
BG_data = zeros(1, 64); % set an empty array for stroring x variable frequency
FG_data = zeros(1, 64);
% using find_second_large function to get x variable
for row = 1 : size(TrainsampleDCT_BG,1)
    idx = find_second_large(TrainsampleDCT_BG(row,:));
    BG_data(idx) = BG_data(idx) + 1;
end

for row = 1 : size(TrainsampleDCT_FG,1)
    idx = find_second_large(TrainsampleDCT_FG(row,:));
    FG_data(idx) = FG_data(idx) + 1;
end
```

To get probability on cheetah, simply divide vector with all cheetah pixel counts. Same for grass.

```
% convert cumulative data into probability
FG_data = FG_data / size(TrainsampleDCT_FG, 1);
BG_data = BG_data / size(TrainsampleDCT_BG, 1);
% display as histogram
subplot(2,1,1);
bar(BG_data);
title('P(x|grass)');
subplot(2,1,2);
bar(FG_data);
title('P(x|cheetah)');
```

Histogram

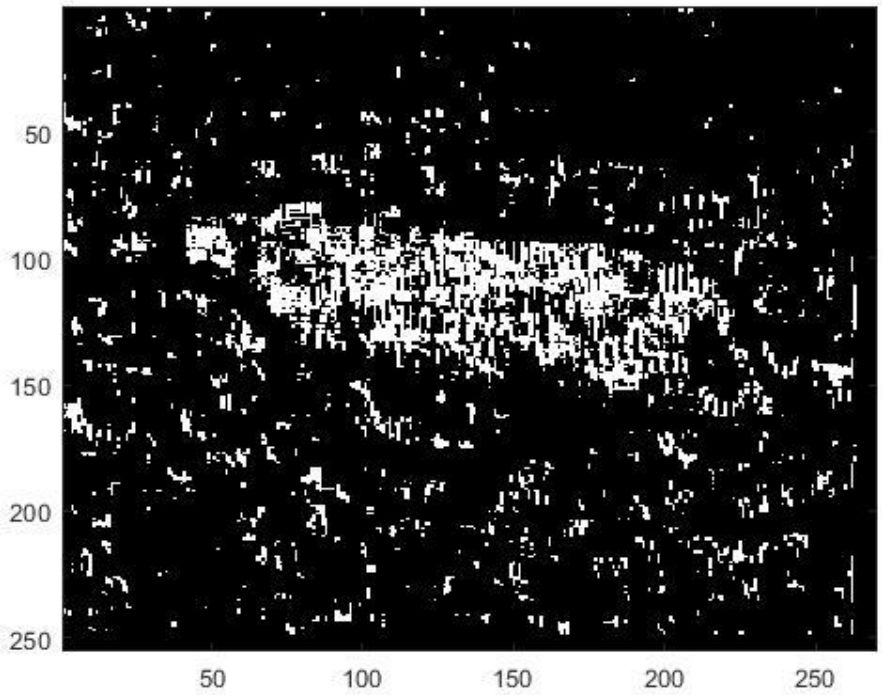



c) According bayes decision rule for 0-1 loss, we need to find i*(x) = argmax[P(x|i)*P(i)] for deciding 0 or 1.

Defining function **expand_zigzag** to help us sort dct2 coefficients.

```
function myArray = expand_zigzag(matrix)
    load("Zig-Zag Pattern.txt");
    myArray = zeros(1, 64);
    for row = 1 : size(matrix,1)
        for column = 1 : size(matrix,2)
            number = Zig_Zag_Pattern(row, column) + 1;
            myArray(number) = matrix(row, column);
        end
    end
end
```

```
% create a new matrix A for storing decision
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A = zeros(row_size, column_size);
% using 8 * 8 blocks to represent the left top pixel
for rows = 1 : row_size - 8 + 1
    for columns = 1 : column_size - 8 + 1
        block = cheetah(rows:rows+7, columns:columns+7);
        block = dct2(block);
        % get X feature
        x_feature = find_second_large(expand_zigzag(block));
        % calculate P(1|x) and P(0|x), find bigger one
        P_0 = BG_data(x_feature) * prior_Pgrass;
        P_1 = FG_data(x_feature) * prior_Pcheetah;
        if (P_0 >= P_1)
            A(rows, columns) = 0;
        else
            A(rows, columns) = 1;
        end
    end
end
% display image
figure(2);
imagesc(A);
colormap(gray(255));
```



Notice: there are edges on the right columns and bottom rows, because there are not sufficient 8 * 8 blocks to represent pixel over there.

d) For estimating probability of error, we need to count how many pixels aren't correct, and then divide it with total pixel counts.

However, I think pixels on the right columns and bottom rows are meaningless, because their value isn't generated by my algorithm. I will remove these pixels in estimating probability of error.

For doing matrix subtraction, we need to make two matrixes with same data type, so I convert ground truth matrix into double and divide its value with 255 to get 1 or 0.

```
57      % load cheetah mask.bmp
58      truth = imread("cheetah_mask.bmp");
59      % calculate last meaningful index of row and column
60      last_row = row_size - 8 + 1;
61      last_column = column_size - 8 + 1;
62      % only take meaningful part
63      truth = double(truth(1 : last_row, 1 : last_column) / 255);
64      A = A(1 : last_row, 1 : last_column);
65      err = truth - A;
66      err = abs(err);
67      probability_error = sum(err,'all') / (used_row_size*last_column);
```

```
Command Window
New to MATLAB? See resources for Getting Started.

>> probability_error

probability_error =

    0.1816

fx >>
```

Ans:
probability of error = 0.1816

## Full Code Review:

```
clear; clc;
% load trainging sample and image
load('TrainingSamplesDCT_8.mat');
cheetah = imread('cheetah.bmp');
% calculate prior probabilities of cheetah and grass
pixel_total_count = size(TrainsampleDCT_FG, 1) + size(TrainsampleDCT_BG, 1);
prior_Pcheetah = size(TrainsampleDCT_FG, 1) / pixel_total_count;
prior_Pgrass = size(TrainsampleDCT_BG, 1) / pixel_total_count;
% calculate P(x|cheetah) and P(x|grass)
BG_data = zeros(1, 64); % set an empty array for stroring x variable frequency
FG_data = zeros(1, 64);
% using find_second_large function to get x variable
for row = 1 : size(TrainsampleDCT_BG,1)
    idx = find_second_large(TrainsampleDCT_BG(row,:));
    BG_data(idx) = BG_data(idx) + 1;
end

for row = 1 : size(TrainsampleDCT_FG,1)
    idx = find_second_large(TrainsampleDCT_FG(row,:));
    FG_data(idx) = FG_data(idx) + 1;
end
% convert cumulative data into probability
FG_data = FG_data / size(TrainsampleDCT_FG, 1);
BG_data = BG_data / size(TrainsampleDCT_BG, 1);
% display as histogram
subplot(1,2,1);
bar(BG_data);
title('P(x|grass)');
subplot(1,2,2);
bar(FG_data);
title('P(x|cheetah)');
% create a new matrix A for storing decision
row_size = size(cheetah, 1);
column_size = size(cheetah, 2);
A = zeros(row_size, column_size);
% using 8 * 8 blocks to represent the left top pixel
for rows = 1 : row_size - 8 + 1
    for columns = 1 : column_size - 8 + 1
        block = cheetah(rows:rows+7, columns:columns+7);
        block = dct2(block);
        % get X feature
        x_feature = find_second_large(expand_zigzag(block));
        % calculate P(1|x) and P(0|x), find bigger one
        P_0 = BG_data(x_feature) * prior_Pgrass;
        P_1 = FG_data(x_feature) * prior_Pcheetah;
        if (P_0 >= P_1)
            A(rows, columns) = 0;
        else
            A(rows, columns) = 1;
        end
    end
end
% display image
```

```
figure(2);
imagesc(A);
colormap(gray(255));
% load cheetah mask.bmp
truth = imread("cheetah_mask.bmp");
% calculate last meaningful index of row and column
last_row = row_size - 8 + 1;
last_column = column_size - 8 + 1;
% only take meaningful part
truth = double(truth(1 : last_row, 1 : last_column) / 255);
A = A(1 : last_row, 1 : last_column);
err = truth - A;
err = abs(err);
probability_error = sum(err,'all') / (used_row_size*last_column);
```

```
function x_index = find_second_large(data)
    for index = 1 : size(data)
        arr = abs(data);
        x_value = max(arr(arr<max(arr)));
        x_index = find(arr == x_value);
    end
end
```

```
function myArray = expand_zigzag(matrix)
    load("Zig-Zag Pattern.txt");
    myArray = zeros(1, 64);
    for row = 1 : size(matrix,1)
        for column = 1 : size(matrix,2)
            number = Zig_Zag_Pattern(row, column) + 1;
            myArray(number) = matrix(row, column);
        end
    end
end
```