

Computer Problem Solution

- a) Using the training data in **TrainingSamplesDCT8.mat**, what are reasonable estimates for the prior probabilities?

Solution:

Two priors probabilities, $P_Y(cheetah)$ and $P_Y(grass)$, could be estimated based on the number of vectors in the training set. The estimation of $P_Y(cheetah)$ and $P_Y(grass)$ are:

$$P_Y(cheetah) = N_{FG}/(N_{FG} + N_{BG}) \quad (1)$$

$$P_Y(grass) = N_{BG}/(N_{FG} + N_{BG}) \quad (2)$$

where

N_{BG} is the number of vectors in matrix **TrainsampleDCT_BG**

N_{FG} is the number of vectors in matrix **TrainsampleDCT_FG**

- b) Using the training data in **TrainingSamplesDCT8.mat**, compute and plot the index histograms $P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$.

Solution:

According to training data, the frequency histograms is the following picture:

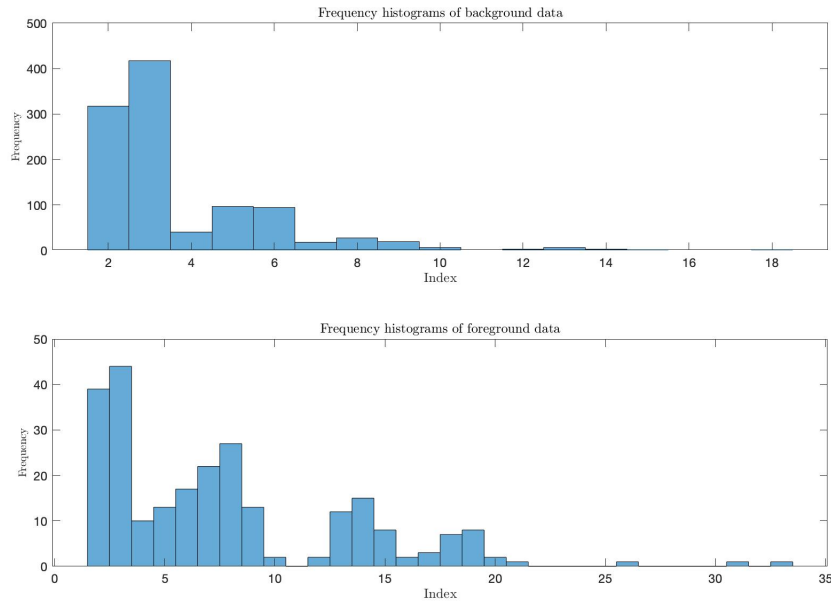


Figure 1: Frequency histograms

The index histograms of $P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ is showed as following:

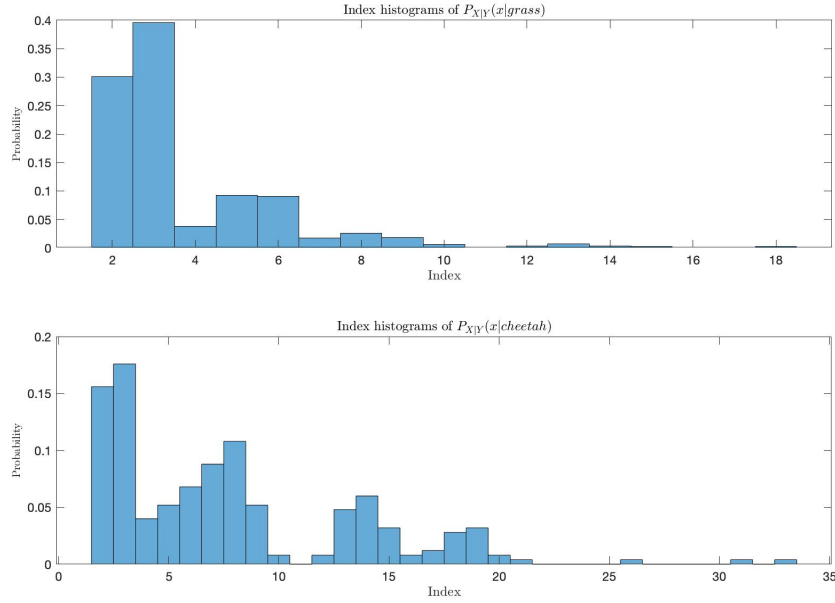


Figure 2: Index histograms

- c) For each block in the image **cheetah.bmp**, compute the feature X (index of the DCT coefficient with 2^{nd} 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:

Given a 8×8 block from the image **cheetah.bmp**, we can easily compute an array of 8×8 frequency coefficients by using function **dct2** on Matlab. Feature X would be index of the 2^{nd} greatest DCT coefficient. Given $X = x$ in one block, we can compute the $P_{Y|X}(cheetah|x)$ and $P_{Y|X}(grass|x)$ as following:

$$P_{Y|X}(cheetah|x) = \frac{P_{X|Y}(x|cheetah) * P_Y(cheetah)}{P_{X|Y}(x|cheetah) * P_Y(cheetah) + P_{X|Y}(x|grass) * P_Y(grass)} \quad (3)$$

$$P_{Y|X}(grass|x) = \frac{P_{X|Y}(x|grass) * P_Y(grass)}{P_{X|Y}(x|cheetah) * P_Y(cheetah) + P_{X|Y}(x|grass) * P_Y(grass)} \quad (4)$$

where

$P_{X|Y}(x|cheetah)$ and $P_{X|Y}(x|grass)$ are the estimation we get from training data.

$P_Y(cheetah)$ and $P_Y(grass)$ are the estimation we get from training data.

According to minimum probability of error rule, if $P_{Y|X}(\text{cheetah}|x) \geq P_{Y|X}(\text{grass}|x)$, then we mask the top left corner of the 8×8 block as 1, regarding this pixel belongs to cheetah. Otherwise, we mask 0. By using a sliding window that moves by one pixel at each step, finally we get a array A containing the mask ndicates which blocks contain grass and which contain the cheetah.

- 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.

Solution:

The comparison between ground truth and picture generated from array A is showed as following:

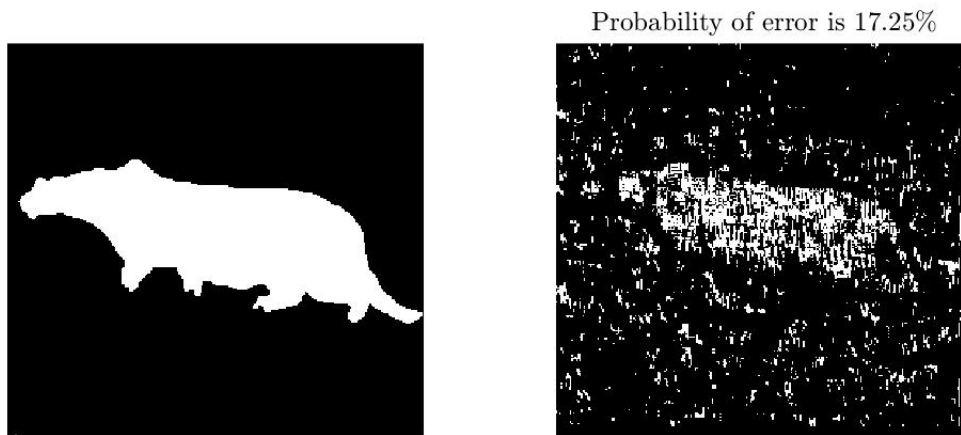


Figure 3: Comparison

The probabilities of error is 17.25%, as showed in the figure above.

Appendix

The following is the Matlab code.

```

1  clear all
2  %%
3  %Training
4  %Read the TrainingSamplesDCT_8.mat file
5  load('dataset/TrainingSamplesDCT_8.mat');
6  %Save TrainsampleDCT_BG and TrainsampleDCT_FG in temporary value
7  train_BG = TrainsampleDCT_BG;
8  train_FG = TrainsampleDCT_FG;
9
10 %Find the second largest value in each row of matrix train_BG
11 [MBG,NBG] = max(train_BG,[],2);
12 train_BG(bsxfun(@eq, train_BG, MBG)) = -1; % Set the largest
    value in each row to -inf
13 [MBG,NBG] = max(train_BG,[],2);
14
15 %Find the second largest value in each row of matrix train_FG
16 [MFG,NFG] = max(train_FG,[],2);
17 train_FG(bsxfun(@eq, train_FG, MFG)) = -1; % Set the largest
    value in each row to -inf
18 [MFG,NFG] = max(train_FG,[],2);
19
20 % [MBG,NBG] = find(train_BG==max(train_BG(:))); % Find the
    largest coefficient
21 % train_BG(MBG,NBG) = -Inf; % Set the largest value as -Inf
22 % [MBG,NBG] = find(train_BG==max(train_BG(:))); % Find the
    second largest coefficient and its position
23 %
24 % [MFG,NFG] = find(train_FG==max(train_FG(:))); % Find the
    largest coefficient
25 % train_FG(MBG,NBG) = -Inf; % Set the largest value as -Inf
26 % [MFG,NFG] = find(train_FG==max(train_FG(:))); % Find the
    second largest coefficient and its position
27
28 %Plot the frequency histogram
29 subplot(2,1,1);
30 h1 = histogram(NBG);
31 ylim([0, 500]);
32 ylabel('Frequency', 'interpreter', 'latex', 'FontSize', 10);

```

```

33 xlabel('Index', 'interpreter', 'latex');
34 title({'Frequency histograms of background data'}, 'FontSize', 12,
    'interpreter', 'latex');
35 subplot(2,1,2);
36 h2 = histogram(N_FG);
37 ylim([0, 50]);
38 ylabel('Frequency', 'interpreter', 'latex', 'FontSize', 10);
39 xlabel('Index', 'interpreter', 'latex');
40 title({'Frequency histograms of foreground data'}, 'FontSize', 12,
    'interpreter', 'latex');
41 %Save the statistic data
42 F_x_BG = zeros(1,64);
43 F_x_BG(min(N_BG):max(N_BG)) = h1.Values;
44 F_x_FG = zeros(1,64);
45 F_x_FG(min(N_FG):max(N_FG)) = h2.Values;
46 %Save the histogram figure
47 set(gcf, 'Position', [400,100,900,600]);
48 saveas(gcf, ['Images/histograms1.jpg']);
49 close(gcf);
50
51 %Calculate the estimation of class-conditionals for two classes
    and priors probabilities
52 P_x_BG = F_x_BG ./ sum(F_x_BG);
53 P_x_FG = F_x_FG ./ sum(F_x_FG);
54 P_BG = size(train_BG,1) / (size(train_BG,1) + size(train_FG,1));
55 P_FG = size(train_FG,1) / (size(train_BG,1) + size(train_FG,1));
56
57 %%Plot the index histogram
58 subplot(2,1,1);
59 h1 = histogram(N_BG, 'Normalization', 'pdf');
60 ylim([0, 0.4]);
61 ylabel('Probability', 'interpreter', 'latex', 'FontSize', 10);
62 xlabel('Index', 'interpreter', 'latex');
63 title({'Index histograms of $$P_{X|Y}(x|grass)$$'}, 'FontSize',
    12, 'interpreter', 'latex');
64 subplot(2,1,2);
65 h2 = histogram(N_FG, 'Normalization', 'pdf');
66 ylim([0, 0.2]);
67 ylabel('Probability', 'interpreter', 'latex', 'FontSize', 10);
68 xlabel('Index', 'interpreter', 'latex');
69 title({'Index histograms of $$P_{X|Y}(x|cheetah)$$'}, 'FontSize',
    12, 'interpreter', 'latex');

```

```
70 %Save the histogram figure
71 set(gcf, 'Position', [400, 100, 900, 600]);
72 saveas(gcf, ['Images/histograms2.jpg']);
73 close(gcf);
74
75 %Read original image
76 I = imread('dataset/cheetah.bmp');
77 I = im2double(I);
78 %Define the loop numbers
79 loop_row = size(I,1) - 8 + 1;
80 loop_column = size(I,2) - 8 + 1;
81
82 mask = zeros(size(I));
83 position_ref = load('dataset/Zig-Zag Pattern.txt');
84 for i=1:1:loop_row
85     for j=1:1:loop_column
86         block = I(i:i+7,j:j+7);
87         DCT_block = dct2(block);
88         DCT_block = abs(DCT_block);
89         [x,y] = find(DCT_block==max(DCT_block(:))); % Find the
            largest coefficient
90         DCT_block(x,y) = -1; % Set the largest value as -1
91         [x,y] = find(DCT_block==max(DCT_block(:))); % Find the
            second largest coefficient and its position
92         feature = position_ref(x,y) + 1;
93         %Decide the binary mask
94         %Before decide the mask, we should caluate two class-
            conditionals
95         P_FG_Decision = P_x_FG(1,feature) * P_FG / (P_x_FG(1,
            feature)* P_FG + P_x_BG(1,feature) * P_BG);
96         P_BG_Decision = P_x_BG(1,feature) * P_BG / (P_x_FG(1,
            feature)* P_FG + P_x_BG(1,feature) * P_BG);
97         % P_FG_Decision = P_x_FG(1,feature) / (P_x_FG(1,feature)*
            P_FG + P_x_BG(1,feature) * P_BG);
98         % P_BG_Decision = P_x_BG(1,feature) / (P_x_FG(1,feature)*
            P_FG + P_x_BG(1,feature) * P_BG);
99         if P_FG_Decision >= P_BG_Decision
100             mask(i,j) = 1;
101         end
102     end
103 end
104
```

```
105 subplot(1,2,1)
106 I = imread('dataset/cheetah_mask.bmp');
107 I = im2double(I);
108 imshow(I);
109 subplot(1,2,2)
110 imshow(mask);
111 %Calculate the probability of error
112 error = length(find((mask-I)~=0)) / (size(I,1) * size(I,2));
113 title(['Probability of error is ', num2str(error*100, '%.2f'), '%',
114        ], 'FontSize', 12, 'interpreter', 'latex');
114 %set(gcf, 'Position', [900, 600]);
115 saveas(gcf, ['Images/segmentation.jpg']);
116 close(gcf);
```