ECE 271A: Statistical Learning I - Quiz #1

Mazeyu Ji - A59023027

December 1, 2023

Question a)

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

Solution:

Let:

- n_{FG} represent the number of foreground (cheetah) samples.
- n_{BG} represent the number of background (grass) samples.

The prior probabilities can be estimated as:

$$P_Y(\text{cheetah}) = \frac{n_{FG}}{n_{FG} + n_{BG}} = 0.1919 \tag{1}$$

$$P_Y(\text{grass}) = \frac{n_{BG}}{n_{FG} + n_{BG}} = 0.8081 \tag{2}$$

Question b)

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:

To compute the index histograms for both the foreground and background, one would tally the occurrences of the second-largest energy value falling into different indices for each 64-dimensional vector. Subsequently, these counts are normalized, with the resulting frequencies serving as the estimated conditional probabilities.

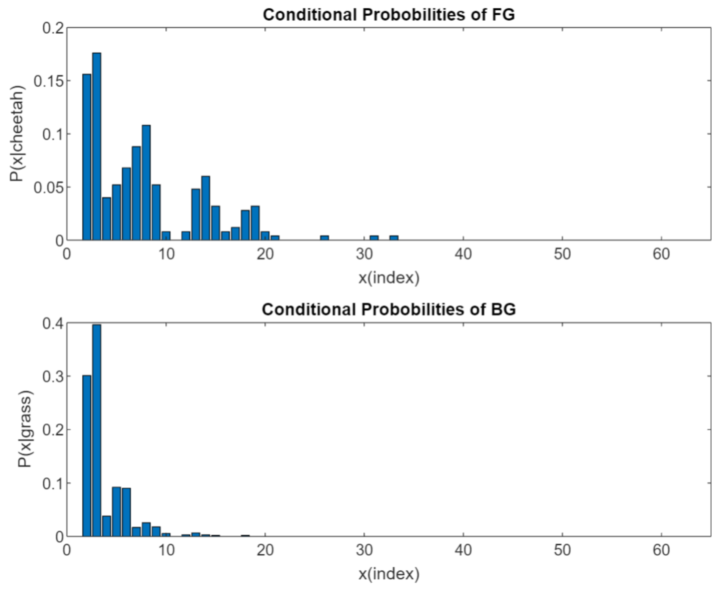

Figure 1: Histograms showing the distribution of the index of the coefficient with the second-largest energy value for both classes. The top histogram represents the conditional probabilities for the cheetah class, $P_{X|Y}(x|\text{cheetah})$, while the bottom histogram depicts those for the grass class, $P_{X|Y}(x|\text{grass})$.

Question 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:

To construct the mask, the top-left pixel of an 8×8 block is marked as '1' if the block is identified as containing the cheetah, otherwise, it is set to '0'. To facilitate the calculation of DCT for boundary pixels, we extend both the length and width of the image and fill the unknown areas with zeros. Implementing a sliding window mechanism that shifts by one pixel at each iteration, the final array A is produced. For a given block, we first compute the index where the second-largest DCT energy value is located. Based on this index and using the minimum probability of error rule, the state can be identified as 'cheetah', expressed as:

$$\frac{P_{X|Y}(x|\text{cheetah})}{P_{X|Y}(x|\text{grass})} > \frac{P_Y(\text{grass})}{P_Y(\text{cheetah})} = T \tag{3}$$

Where:

- $P_{X|Y}(x|\text{cheetah})$ and $P_{X|Y}(x|\text{grass})$ are class conditional probabilities estimated from the training data.
- $P_Y(\text{cheetah})$ and $P_Y(\text{grass})$ represent the prior probabilities estimated from the training data.
- T denotes the decision threshold.

The generated mask is shown below:

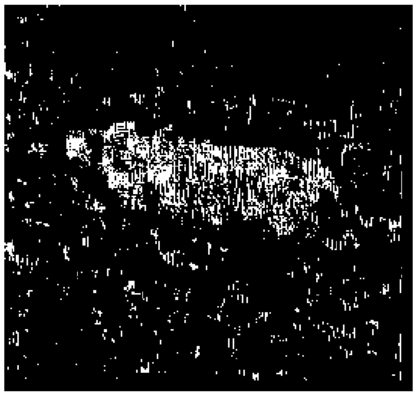

Figure 2: Generated mask.

Question 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** and compute the probability of error of your algorithm.

Solution:

To determine the probability of error, we conducted a pixel-by-pixel comparison between the mask generated by our algorithm and the ground truth. We then gauged the frequency of misclassifications by calculating the proportion of incorrectly classified pixels to the total pixels. Mathematically, this can be represented as:

$$P_{\text{error}} = \frac{e}{T} = 16.90\% \tag{4}$$

Where:

- P_{error} is the probability of error.
- e represents the number of misclassified pixels.
- T is the total number of pixels.

Appendix

```
%% Estimate the prior probabilities
load('TrainingSamplesDCT_8.mat')
[rowFG, columnFG] = size(TrainsampleDCT_FG);
[rowBG, columnBG] = size(TrainsampleDCT_BG);
priorFG = rowFG / (rowBG + rowFG);
priorBG = rowBG / (rowBG + rowFG);
%% Compute and plot index histograms
% Compute the conditional probabilities
indexBinFG = zeros(columnFG,1);
for i = 1:rowFG
    row = TrainsampleDCT_FG(i, :);
    [~, sortedIndices] = sort(row, 'descend');
    indexBinFG(sortedIndices(2)) = indexBinFG(sortedIndices
        (2)) + 1;
end
conditionalProbsFG = indexBinFG / sum(indexBinFG(:));

indexBinBG = zeros(columnBG,1);
for i = 1:rowBG
    row = TrainsampleDCT_BG(i, :);
    [~, sortedIndices] = sort(row, 'descend');
    indexBinBG(sortedIndices(2)) = indexBinBG(sortedIndices
        (2)) + 1;
end
conditionalProbsBG = indexBinBG / sum(indexBinBG(:));

% Plot the histograms
figure;

subplot(2, 1, 1);
indicesFG = 1:length(conditionalProbsFG);
bar(indicesFG, conditionalProbsFG);
xlabel('x(index)');
ylabel('P(x|cheetah)');
title('Conditional Probobilities of FG');
xlim([min(indicesFG)-1, max(indicesFG)+1]);

subplot(2, 1, 2);
indicesBG = 1:length(conditionalProbsBG);
bar(indicesBG, conditionalProbsBG);
xlabel('x(index)');
ylabel('P(x|grass)');
title('Conditional Probobilities of BG');
xlim([min(indicesBG)-1, max(indicesBG)+1]);

%% Compute DCT coefficient and separate the foreground and
    background
% Read and preprocess the image
```

```
img = imread('cheetah.bmp');
imgDouble = im2double(img);
[height, width] = size(imgDouble);
extendedImg = zeros(height + 7, width + 7);
extendedImg(1:end-7, 1:end-7) = imgDouble;

% Classify BG and FG
pattern = readmatrix('Zig-Zag Pattern.txt') + 1;
indexMap = zeros(64, 1);% Map index for accelerated
    algorithm speed
for i = 1:8
    for j = 1:8
        indexMap(j + (i-1)*8) = pattern(i, j);
    end
end

predictedMask = zeros(height, width);
thresStar = priorBG / priorFG;
for i = 1:height
    for j = 1:width
        block = extendedImg(i:i+7, j:j+7);
        dctBlock = abs(dct2(block));
        % Find the index of the second largest value
        dctBlock = dctBlock';
        dctLine = dctBlock(:); % Flatten the matrix row by
            row
        [~, index] = sort(dctLine, 'descend');
        indexMapped = indexMap(index(2));
        % Calculate the probability
        thres =conditionalProbsFG(indexMapped) /
            conditionalProbsBG(indexMapped);
        if thres > thresStar
            predictedMask(i, j) = 1;
        end
    end
end

figure;
imagesc(predictedMask);
colormap(gray(255));

%% Compute error
maskGT = imread('cheetah_mask.bmp');
maskGT = im2double(maskGT);
error = sum(sum(predictedMask ~= maskGT)) / (height * width)
```