EEE 431 (FALL 2021)

MATLAB ASSIGNMENT 1 (Due: November 11, 2021, 17:00)

The objective of this MATLAB assignment is to study a number of issues with uniform/non-uniform scalar quantization, Huffman (de)coding and Lempel-Ziv (de)coding. *It is not allowed to use built-in functions of MATLAB for quantization, Huffman (de)coding and Lempel-Ziv (de)coding.*

Part 1 (70 points)

Consider a random source that emits independent and identically distributed (i.i.d.) samples, and let Y denote the output of this source. Take the two non-zero rightmost digit of your student number and call these digits a and b from left to right. For instance, if your student number is 20000403, then $a = 4$ and $b = 3$. Let X_1 be a random variable uniformly distributed in the interval $[0, a]$, X_2 be a random variable uniformly distributed in the interval $[-b, 0]$, and X_1 and X_2 are independent. Let $Y = X_1 + X_2$.

In the following, various quantization techniques are employed to quantize the samples generated by this source.

1) Obtain the PDF of Y and calculate the average source power $\mathbb{E}[Y^2]$. (No MATLAB is used in this part; obtain the results theoretically.)

2) Start your MATLAB script with the following lines: close all; clear all; rand('seed', sum(100*clock));

3) Generate a long sequence of source samples Y (1000000 samples) and plot its histogram (with sufficient resolution). Also calculate and report the average value of the squares of these samples and compare it with the theoretical result in part 1) above.

4) Then, calculate the quantized versions of Y (denoted by $\tilde{Y}$) by considering $N = 8$ levels in the uniform quantizer. Calculate the sequence of the quantization errors corresponding to the source samples you generated (Note that the quantization error is defined as $e = Y - \tilde{Y}$.). Calculate and report the average power of the quantization error and the resulting signal-to-quantization-noise ratio (SQNR) in dB (i.e., 10 log(SQNR)).

5) Determine the probability of each possible quantization output based on their frequency of occurrence, and plot the probability mass function

(PMF) of the possible quantization outputs. Then, based on these probabilities, design a Huffman code both in Matlab and theoretically. Your Matlab code should take the probabilities as input and output the codewords. List all the codewords and calculate the average codeword length. Then, apply this Huffman code to the sequence of quantization outputs in Matlab. Compare the average codeword length with the case of no coding.

6) Write a code for Huffman decoding based on the encoding algorithm in part 5). Input the encoded sequence in part 5) to this decoder and show that the output of the decoder is the original input sequence.

Next, a non-uniform quantization is employed instead of a uniform quantization. In particular, a compander (compressor/uniform quantizer/expander) as described in the class is used. Here, instead of standard companders such as μ-law compander a method designed for the source at hand is employed.

7) Obtain the CDF of Y denoted by $F(x)$. Also, obtain the inverse of CDF, i.e., $F^{-1}(x)$. (No MATLAB is used in this part; obtain the results theoretically.)

8) Implement $F(x)$ and $F^{-1}(x)$ on MATLAB.

9) Implement a non-uniform PCM quantizer with compressor $g(x) = F(x)$ and expander $g^{-1}(x) = F^{-1}(x)$ and a uniform quantizer with $N = 8$ bins.

10) Plot the quantization regions and the corresponding reconstruction points on MATLAB, i.e., you should plot y vs. $Q(y)$ where $Q(y)$ denotes reconstruction point for the input y.

11) Apply this quantizer to the source samples you generated. Calculate the sequence of the quantization errors corresponding to the source samples you generated. Calculate and report the average power of the quantization error and the resulting signal-to-quantization-noise ratio (SQNR) in dB. Compare the results with part 4) above.

12) Determine the probability of each possible quantization output based on their frequency of occurrence, and plot the probability mass function (PMF) of the possible quantization outputs. Then, based on these probabilities, design a Huffman code both in Matlab and theoretically. Your Matlab code should take the probabilities as input and output the codewords. List all the codewords and calculate the average codeword length. Then, apply this Huffman code to the sequence of quantization outputs in Matlab. Compare the average codeword length with the case of no coding.

13) Write a code for Huffman decoding based on the encoding algorithm in part 12). Input the encoded sequence in part 12) to this decoder and show that the output of the decoder is the original input sequence.

Next, Lloyd-Max algorithm is employed to obtain a quantizer.

14) Design a 8-level quantizer for this source using the Lloyd-Max algorithm. Determine and report the boundaries of the quantization regions and the reconstruction levels.

15) Plot the quantization regions and the corresponding reconstruction points on MATLAB. Compare this plot with the plot in part 10) above.

16) Quantize the sequence of Y values you generated with the quantizer you obtained in part 14) above. Calculate the sequence of the quantization errors corresponding to the source samples you generated. Calculate and report the average power of the quantization error and the resulting signal-to-quantization-noise ratio (SQNR) in dB (i.e., 10 log(SQNR)). Compare the results with part 13) above.

Part 2 (30 points)

In this part, we will perform Lempel-Ziv coding and decoding. Take a long English text with around 500 words (e.g., from Wikipedia). Convert all the lower case letters to uppercase letters (you can use 'upper' command of MATLAB). Remove any character that does not belong to the English alphabet except for the space character (you can use 'strrep' command of MATLAB). Append '#' to the end of this long sequence of characters. The possible characters for coding purposes are all the capital letters in the English alphabet, the space character and '#' character which denotes the termination symbol.

1) Construct an initial dictionary with entries # $\rightarrow$ 0, $A \rightarrow 1$, $B \rightarrow 2$, ..., $Z \rightarrow 26$, SPACE CHARACTER $\rightarrow$ 27. Perform Lemple-Ziv encoding on the sequence of characters as described in the class. Your code should output a binary sequence. When converting the output integers to a binary sequence, take the current size of the dictionary into account. For instance, an output integer 20 must be converted to '00010100' when the current size of the dictionary is 140 and to '010100' when the current length of the dictionary is 32. For conversion from decimal to binary, you are allowed to use a built-in function of MATLAB.

2) Calculate and report the average codeword length. Compare the average codeword length with the case of no coding.

3) Take the binary sequence from part 1) and perform Lempel-Ziv decoding on this sequence. Note that you should start from the initial dictionary with entries # $\rightarrow$ 0, $A \rightarrow 1$, $B \rightarrow 2$, ..., $Z \rightarrow 26$, SPACE CHARACTER $\rightarrow$ 27. Show that you recover the initial character sequence.

4) Determine the probability of each possible character based on their frequency of occurrence, and plot the probability mass function (PMF) of

the possible characters. Then, based on these probabilities, compute the entropy of the sequence. Compare the average codeword length part 2) above with the entropy you computed here. Comment on this comparison. What do you expect if you take a longer or shorter sequence of characters?

Technical/Reporting Requirements:

Note that you are not allowed to use the built-in functions from Matlab (or, other resources) to complete the project. You must write your own code, and conduct your simulations using that code. Both the Matlab files and the project reports will be processed by turnitin. In addition, the m files will be checked via the Moss software.

Your report should contain all the relevant information about the set-up used (the specific schemes implemented, parameters selected, etc), results obtained and your comments on the results. The specific format is up to you, but please make sure to properly label each figure, include relevant captions, point to the right results in your explanations, etc. It should include a title page, brief introduction and outline as well as any references used. The references used should be cited within the report wherever they are used.

The report must be typed using an advanced wordprocessor (e.g. latex, word, etc), and should be submitted as a pdf file on the course Moodle site. Please also submit your Matlab codes as a separate (single) file. *The submission links for the pdf report and the Matlab m file will be separate.*

The file name format is LastName-FirstName.pdf or LastName-FirstName.m