Lab 2 : Fundamentals of Information Theory

Objective

- To understand the basic ideas of source entropy, source coding.
- To compute the source entropy using MATLAB.
- To learn source coding and decoding using Huffman code.

I. Entropy

A. Introduction

The entropy of a discrete random variable X is a function of its probability mass function (PMF). If X has alphabet $\mathcal{A} = \{a_1, \cdots, a_N\}$ and respective probabilities $\{p_1, \cdots, p_N\}$. The entropy of X is defined as

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i.$$

Entropy provides a quantitative measure of the information content of a source modeled by X.

B. Lab Problem

Develop a computer program that allows you to calculate and plot the entropy of a source with variable probabilities.

1) For a binary source with alphabet $\{x_1, x_2\}$ and respective probabilities $\{p, 1-p\}$, plot the entropy as a function of p. When will the entropy takes its maximum value and what is the maximum value?
2) Consider a ternary source with alphabet $\{x_1, x_2, x_3\}$ and respective probabilities p, q, 1 p q . Write a program to find out when will the entropy takes its maximum value and what is the maximum value.

II. Source Encoding and Decoding Using Huffman Code

A. Introduction

Huffman coding is a source coding method to design codes whose rate is close to the source entropy. The procedure of Huffman code design was mentioned in detail in class. The basic idea is to use long codewords for less frequent letters and short codewords for more frequent letters for data compression. The average codeword length of a Huffman code can be calculated as

$$\bar{l} = \sum_{i=1}^{N} p_i l_i,$$

where l_i is the length of the codeword for the ith letter and p_i is the probability corresponding to that letter. If the source has entropy $H(X)$, the efficiency of the Huffman code is

$$\eta = \frac{H(X)}{\bar{l}}.$$

In this lab, you will be asked to conduct source encoding and decoding of a file containing English letters using Huffman code; and observe the Huffman codewords and the compression rate. For simplicity, the file only contains letters and space, i.e, $\mathcal{A} = \{A, B, C, \cdots, Y, Z, a, b, c, \cdots, y, z, \text{Space}\}$. For the design of the Huffman code and the encoding/decoding, you can use existing functions in MATLAB.

B. Simulation Procedure

1) Use *fread* to read data from the file *source.txt* and save it as an array. Make sure to save the data in 8-bit unsigned integer data type (use *uint8* function). You can display it on screen to see that it contains letters and space.
2) Find the probability of each symbol in the file by counting the number of occurrences of each symbol and divide it by the total number of symbols in the file. You can calculate the entropy based on the probabilities.
3) Construct a Huffman code using MATLAB function *huffmandict*. Use *help* function to get more details about the syntax of this function. Display the different codewords and observe the relation between the codeword length and the probability of the corresponding symbol.
4) Encode each symbol in the data array using the codewords you obtained from the previous step using the *huffmanenco* function.
5) Divide the obtained binary bit-stream into blocks of 8-bits and convert it into decimal numbers. Write the decimal numbers into the new file *encode.huf* using *fwrite*. The new file is a binary file not necessarily a text file. Compare the compressed file size and the original file size. You have to add '0' to the end of the array if the its length cannot be divided by 8. The following partial code may help.

```
comp = huffmanenco(A,dict);
L = 8-rem(length(comp),8);
b1 = [comp' zeros(1,L)];
```

6) Use *fread* again to read data from the compressed file *encode.huf* and save it in an array. Convert the data from decimal numbers into binary bit-stream.
7) Recover the original symbols from the binary bit stream using the *huffmandeco* function and save it into a new file *decode.txt* using the *fprintf* function. Note that you should write the symbols into a text file. Compare the original file and your decoded file. The decoding step is more difficult and it will take longer time than the encoding step.

C. Lab Problem

Huffman Code for Printed English (Simplified). Follow the provided procedure to encode the provided file *source.txt* into *encode.huf* using Huffman code; and decode the compressed file into *decode.txt*. Answer the following questions.

1) List your Huffman codeword for each source symbol and display the probability of each symbol as well.
2) Calculate the efficiency of the Huffman code you use.
3) What is the compression rate (ratio of the size of the encoded file to the original file)?
4) Is your decoded file the same as the original file?

Lab Report Guide

Name:

Student ID:

Lab Report Requirements:

- Grade of late report will be 10% off based on regular.
- Lab reports should meet the requirements posted on the eClass.

Lab Marking Guide:

About lab report	
Abstract	/2
Introduction	/2
Matlab codes	/10
Figures, results and comments	/16
Organization, content, and clarity	/10
Total	/40