**Lecture 12 Information Theory**

**LEARNING OUTCOME**

**By the end of this lesson a student will be able to:**

1. understand a concept of an entropy in information theory
2. build Huffman tree and encoding process on given symbols and pdf
3. compute an entropy from a probability density function

**Random Variable**

Information entropy is defined as an average practical amount of information produced by a random variable. Random variable is not random nor variable. It is a function assigned to an event. Suppose we flip or toss a fair coin.



A fair coin is tossed will produce P(X=1) = 0.5 and P(X=0) = 0.5 ideally.

An event head is facing up or tail is facing. The random variable is a number assigned to an event. A random variable X: head→1 and X: tail→0. In photography, computing, and colorimetry, a grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. The 8-bit gray level is assigned from 0 as black to 255 as white.

**Entropy**

Given the probability density function of a random variable, the quest is the entropy. Let Y be gender of a population in Malaysia. P(Y=1) = 0.48 and P(Y=0) = 0.52.

What is the amount of information carried by the random variable Y? The measure of information entropy carried X= *x*0, *x*1, …, *xn*−1 with the probability *p*0, *p*1, …, *pn*−1, an entropy measure



In a real life in Malaysia, it is popular to use 6-digit ATM PIN numbers. There are 1 million combinations. They are not randomly picked nor uniformly distributed.

**Huffman Code**

Huffman encoding is the most basic variable length coding method. Others are Arithmetic Coding, Lempel-Ziv Algorithm, Run Length Encoding(RLE), and soon. LZW algorithm has been used in GIF and ZIP. Arithmetic Coding is 10% better or more efficient than Huffman coding. Suppose we have some symbols to encode. The issue is here how in the most efficient way to encode the given symbols in binary. The objective is to represent the symbols using the minimum number of bits.

Table 1: A probability density function of given symbols A-J.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | A | B | C | D | F | G | H | I | J |
| *P*(*X=x*) | 0.021 | 0.120 | 0.238 | 0.264 | 0.200 | 0.103 | 0.037 | 0.015 | 0.002 |

Let us draw the probability distribution

Figure 1. A typical Erlang probability density distribution is skewed to the right

In Huffman encoding strategy,

Step 1: Sort the symbols in descending order according the frequency/probability distribution.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | A | B | C | D | F | G | H | I | J |
| *P*(*X=x*) | 0.021 | 0.120 | 0.238 | 0.264 | 0.200 | 0.103 | 0.037 | 0.015 | 0.002 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | H | A | I | J |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.037 | 0.021 | 0.015 | 0.002 |

Step 2: Combine the last 2 symbols into one node and add the frequency/probability.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | H | A | I | J |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.037 | 0.021 | 0.015 | 0.002 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | H | A | I-J |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.037 | 0.021 | 0.017 |

I

0.015

J

0.002

I-J

0.017

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | H | A | I-J |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.037 | 0.021 | 0.017 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | H | A-IJ |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.037 | 0.038 |

I

0.015

J

0.002

I-J

0.017

A

0.021

A-IJ

0.038

Sorting…

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | A-IJ | H |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.038 | 0.037 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G | AJ-H |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.103 | 0.075 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | B | G-AH |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.120 | 0.178 |

Sorting….

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | G-AH | B |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.178 | 0.120 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | D | C | F | GH-B |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.298 |

Sorting….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | GH-B | D | C | F |
| *P*(*X=x*) | 0.298 | 0.264 | 0.238 | 0.200 |

Sorting….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | GH-B | D | C | F |
| *P*(*X=x*) | 0.298 | 0.264 | 0.238 | 0.200 |

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | GH-B | D | C-F |
| *P*(*X=x*) | 0.298 | 0.264 | 0.438 |

Sorting….

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | C-F | GH-B | D |
| *P*(*X=x*) | 0.438 | 0.298 | 0.264 |

|  |  |  |
| --- | --- | --- |
| Symbol | C-F | GB-D |
| *P*(*X=x*) | 0.438 | 0.562 |

Sorting….

|  |  |  |
| --- | --- | --- |
| Symbol | GB-D | C-F |
| *P*(*X=x*) | 0.562 | 0.438 |

|  |  |
| --- | --- |
| Symbol | GD-CF |
| *P*(*X=x*) | 1.000 |

Then repeat the 2 steps until there is one root node.

Now, we are building the Huffman tree.

GD-CF

1.000 GD-CF

P(X=x) 1.000

GB-D

0.562 GD-CF

P(X=x) 1.000

C-F

0.438 GD-CF

P(X=x) 1.000

|  |  |
| --- | --- |
| Symbol | GD-CF |
| *P*(*X=x*) | 1.000 |

|  |  |  |
| --- | --- | --- |
| Symbol | GB-D | C-F |
| *P*(*X=x*) | 0.562 | 0.438 |

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | C-F | GH-B | D |
| *P*(*X=x*) | 0.438 | 0.298 | 0.264 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | GH-B | D | C | F |
| *P*(*X=x*) | 0.298 | 0.264 | 0.238 | 0.200 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbol | D | C | F | G-AIJH | B |
| *P*(*X=x*) | 0.264 | 0.238 | 0.200 | 0.178 | 0.120 |

GD-CF

1.000 GD-CF

P(X=x) 1.000

GB-D

0.562

C-F

0.438

GH-B

0.298

D

0.264

C

0.238

F

0.200

GH

0.178

B

0.120

G

0.103

AH

0.075

AJ

0.038

H

0.037

A

0.021

IJ

0.017

I

0.015

J

0.002

Then we can assign the binary string to each symbol. Going down the Huffman tree from the root to the left child will get zero and the right child will be assigned as one.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x* | Symbol | *P*(*X=x*) | Huffman Code | Length *nx* | E{X}= *nx*⋅*P*(*X*=*x*) |
| 0 | A | 0.021 | 000100 | 6 | 0.126 |
| 1 | B | 0.120 | 001 | 3 | 0.360 |
| 2 | C | 0.238 | 10 | 2 | 0.476 |
| 3 | D | 0.264 | 01 | 2 | 0.528 |
| 4 | F | 0.200 | 11 | 2 | 0.400 |
| 5 | G | 0.103 | 0000 | 4 | 0.412 |
| 6 | H | 0.037 | 00011 | 5 | 0.185 |
| 7 | I | 0.015 | 0001010 | 7 | 0.105 |
| 8 | J | 0.002 | 0001011 | 7 | 0.014 |
| Total |  | 1.0000 |  |  | 2.606 |

Then, we have an average length of Huffman code for the nine symbols is 2.606 bits.

Let us review the entropy of the symbols.

|  |  |  |  |
| --- | --- | --- | --- |
| *x* | Symbol | *px =P*(*X=x*) | *px* log2 1/*px* |
| 0 | A | 0.021 | 0.117043 |
| 1 | B | 0.120 | 0.367067 |
| 2 | C | 0.238 | 0.492890 |
| 3 | D | 0.264 | 0.507247 |
| 4 | F | 0.200 | 0.464386 |
| 5 | G | 0.103 | 0.337766 |
| 6 | H | 0.037 | 0.175984 |
| 7 | I | 0.015 | 0.090883 |
| 8 | J | 0.002 | 0.017932 |
| Total |  | 1.0000 | 2.571200 |

We can see that the average bit length of Huffman code is 2.606. How do we compare the Huffman code performance to entropy of the symbols?

The average bit on Huffman code is near to the entropy. We will consider the Huffman encoding scheme perform well. The next popular encoding scheme is arithmetic coding. It is slightly more efficient than Huffman.

Sketch the graph of probability distribution

Encode the symbols using Huffman technique

Build the Huffman tree

Compute the average bit length of Huffman code on the symbols

Compute the entropy on the symbols

Compare the Huffman encoding scheme performance relative to the entropy of the symbols

Previous Tutorial 11:

Take y = the last 2 digit of an ID number

Let a symbol or random variable X from 0 to 8 follow poisson distribution with  = 3+y/100

Tutorial 12: Information Theory

1. Collect an article or a passage of at least 2000 characters.

2. Count each alphabet and space character.

2. Draw an frequency plot or histogram on alphabets and space.

3. Convert its frequency into a sample pdf

4. Build Huffman Tree on the passage

5. Encode each character as a binary string

6. Compute an average bit length of symbols

7. Compute an entropy on each symbols from the passage.

Why M1 Money Supply (Cash) is Skyrocketing Like No Time History

December 27, 2020

David Haggith

The Great Recession Blog

This is a massive amount of new cash money — historically massive — done almost covertly in the quickest burst ever — and yet it did not even cause the stock market to blink!

Why did such an enormous surge in money supply happen in the last two weeks of November with no financial articles being written about it and no statements from the Fed about it? What is going on behind the scenes at the Fed and/or US treasury right now?

Some advisers say they’re busier than ever in the last weeks of 2020, especially with helping clients transfer wealth to the next generation tax-free while they still can. Appraisers, who are crucial for valuing assets used in these estate planning strategies, have been inundated…

Naturally, the running of the rich is going to create some large temporary piles of cash as money moves from one kind of asset into whatever the rich are planning to do with all this money. (Maybe give it to their entitled children before estate taxes rise again.

So, the rich are rushing to cash out. Though the article mentions nothing about the biggest rise in the nation’s history in supply of the most liquid forms of cash, I suspect the rush of the rich to cash out assets now has something to do with the rapid rise in cash balances. In that case, it may just be temporary, as other assets become purchased with the gains down the road.

If it continues to be held in M1 cash form, then, being easier to spend and move down the road, it could create the kind of inflation I warned about where there is too much (cash) money chasing too few goods during a time of COVID shortages. We’ll have to remain vigilant to see. It is a matter of how the cash circulates.

The amount of dollars in checking accounts in the US banking system exploded by 25% from Nov. 16-30, the fastest rate in history, including the immediate post COVID-19 printing bonanza. The money, an unbelievable $1.3 trillion, came from savings accounts.