Case Study of Huffman Coding

Bachelor of Technology
In
Electronics & Communication Engineering

By

Guna Sai Kiran Nekkanti **Roll No. : BT20ECE075**

Under the Guidance of
Dr. Rashmi Pandhare

Indian Institute of Information Technology, Nagpur

Coding Techniques

I

Details

- **Name :** Nekkanti Guna Sai Kiran
- **Enrollment No :** BT20ECE075
- **Lab Experiment:** 02
- **Course Title :**Coding Techniques
- **Lab Name :**Case Study On Huffman Coding In Information theory
- **Date :**16/02/2023

Case Study on Huffman Coding

I

Introduction to Lossless data Compression

2.1.1 Information Theory And Source Coding

The most significant feature of the communication system is its unpredictability or uncertainty. The transmitter transmits at random any one of the pre-specified messages. The probability of transmitting each individual message is known. Thus our quest for an amount of information is virtually a search for a parameter associated with a probability scheme. The parameter should indicate a relative measure of uncertainty relevant to the occurrence of each message in the message ensemble.

The principle of improbability (which is one of the basic principles of the media world)–"if a dog bites a man, it's no news, but if a man bites a dog, it's a news"—helps us in this regard.

Hence there should be some sort of inverse relationship between the probability of an event and the amount of information associated with it. The more the probability of an event, the less is the amount of information associated with it, vice versa.

I(xj)=f(1/p(Xj)),
Where Xj is an event with a probability p(Xj) the amount of information associated with it is I(xj).
Now let there be another event Yk such that Xj Yk are independent . Hence probability of the joint event is p(Xj, Yk)= p(Xj) p(Yk) with associated information content ,

I(Xj, Yk)=f(1/p(Xj, Yk))= f(1/p(Xj) (Yk))

The total information I(Xj, Yk) must be equal to the sum of individual information I(Xj) I(Yk), where I(Yk)=f(1/p(Yk)).

Thus it can be seen that function f must be a function which converts the operation of multiplication into addition.

LOGARITHM

It is one such function. Thus, the basic equation defining the amount of information (or self-information) is, I(Xj) = log(1/P(Xi)= - log(P(Xi)
when base is 2 (or not mentioned)the unit is bit, when base is e the unit is nat, when base is 10 the unit is decit or Hertley.

ENTROPY:

Entropy is defined as the average information per individual message.
Let there be L different messages m1, m2…….mL, with their respective probabilities of occurrences be p1, p2…….pL. Let us assume that in a long time interval, M messages have been generated. Let M be very large so that M»L The total amount of information in all M messages.

The number of messages m1=M*p1, the amount of information

in message m1= log2(1/P(Xi)), thus the total amount of information in all m1 messages = M*p1* log2(1/P(Xi)).

So, the total amount of information in all L messages will then be I= M*(P1)*log(1/P1)+ M*(P2)*log(1/P2) +……+ M*(PL)*log(1/PL)

So, the average information per message, or entropy, will then be H=I/M=(P1)*log(1/P1)+ (P2)*log(1/P2) + ……+ (PL)*log(1/PL);
Hence, H(X) = - P(Xi) log2(P(Xi)) , summation i=1 to L

The Entropy of a source in bits/symbol is given by H(X) = - P(Xi) log2(P(Xi)) log2L , summation i=1 to L Where Xi are the symbols with probabilities P(Xi) , i=1,2,3 L
The equality holds when the symbols are equally likely.

There are two types of code possible:
1) Fixed Length Code : All code words are of equal length
2) Variable Length Code: All code words are not of equal length. In such cases, it is important for the formation of uniquely decodable code that all the code words satisfy the PREFIX CONDITION, which states that "no code word forms the prefix of any other code word".

The necessary sufficient condition for the existence of a binary code with code words having lengths n1 n2 ..... nL that satisfy the prefix condition is,
2power(-nk) 1 , summation k = 1… ..... L ,

Source Coding Theorem:

Let X be ensemble of letters from a discrete memory less source with finite Entropy H(X) output symbols Xi with probabilities P(Xi) , i=1,2,3… L
It is possible to construct a code that satisfies the prefix condition has an average length R that satisfies the following inequality,

H(x) R < H(x)+1, the efficiency of the prefix code is defined as
= H(x)/R,
where, H(X) = - P(Xi) log2(P(Xi)) ,summation i=1 to L

R = - ni log2(P(Xi)) summation i=1 to L

Here ni denotes the length of ith code word

The source coding theorem tells us that for any prefix code used to represent the symbols from a source, the minimum number of bits required to represent the source symbols on an average must be at least equal to the entropy of the source. If we have found a prefix code that satisfies R=H(x) for a certain source X, we must abandon further search because we can not do any better. The theorem also tells us that a source with higher entropy (uncertainty) requires on an average, more number of bits to represent the source symbols in terms of a prefix code.

Proof:

Lower bound:
First consider the lower bound of the inequality. For codewords that have length nk , 1 k L, the difference H(x)-R can be expressed as

- H(x)-R = P(Xk) log2(1/P(Xk)) - P(Xk) nk , summation K=1 L
- H(x)-R = P(Xk) log2(2power(- nk)/P(Xk)) , summation k=1 L
- H(x)-R (log2e) P(Xk) (2power(- nk)/P(Xk))-1 , summation k=1.....L
- H(x)-R (log2e) (2power(- nk) – 1 , summation k=1 L
- H(x)-R 0 (using KRAFT'S INEQUALITY)
- H(x) R

Upper bound:

Let us select a code word length nk such that

2power(- nk) P(Xk) < 2power(- nk + 1)

First consider, 2power(- nk) P(Xk)

- 2power(-nk) P(Xk) = 1 , summation k=1 L
- log2(P(Xk)) < (- nk + 1)
- nk < 1 - log2(P(Xk))
- P(Xk) nk < P(Xk) + P(Xk) log2(P(Xk)) , summation k=1 L
- R < H(x)+1

II

Huffman Coding

2.2.1 Huffman Coding Algorithm

Huffman coding is an efficient source coding algorithm for source symbols that are not equally probable. A variable length encoding algorithm was suggested by Huffman in 1952, based on the source symbol probabilities P(xi), i=1,2…….,L

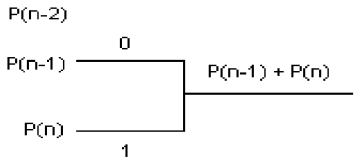
Huffman coding is a lossless data compression algorithm. The main principle in Huffmans coding is that each message in encoded using individual or combination of 0 or 1 bits. Here the most frequent characters will be assigned with the smallest code and the least frequent character will have the largest code. Hence Huffman's coding is based on the variable length approach. Also in Huffman's there is no repetition of the code for the same message.

Data compression has become a necessity while processing information that occupies huge memory. Data compression not only reduces the size but also helps to increase the processing time. Huffman's coding is based on the frequency of occurrence of a data item. The principle is to use a lower number of bits to encode the data that occurs more frequently. Codes are stored in a code book. In all cases the code book plus encoded data must be transmitted to enable decoding.

The algorithm is optimal in the sense that the average number of bits required to represent the source symbols is a minimum

provided the prefix condition is met. The steps of Huffman coding algorithm are given below:

- Arrange the source symbols in increasing order of heir probabilities.
- Take the bottom two symbols tie them together .Add the probabilities of the two symbols write it on the combined node. Label the two branches with a "1 a "0"



- Treat this sum of probabilities as a new probability associated with a new symbol. Again pick the two smallest probabilities, tie them together to form a new probability. Each time we perform the combination of two symbols we reduce the total number of symbols by one. Whenever we tie together two probabilities (nodes) we label the two branches with a "0 a "1 .
- Continue the procedure until only one procedure is left (it should be one if your addition is correct). This completes the construction of the Huffman Tree.
- To find out the prefix codeword for any symbol, follow the branches from the final node back to the symbol. While tracing back the route read out the labels on the branches. This is the codeword for the symbol.

III

An Optimized Huffman's Coding by the method of Grouping

To show this method of Huffman's coding, we have used text as an example. Consider the below Text example of Huffman's coding.

"IEEECOMPUTATIONALINTELLIGENCE"

Here in the above text we have 29 letters. The frequency and character is

Character	Frequency
E	6
I	4
T	3
N	3
L	3
C	2
O	2
A	2
M	1
P	1
U	1
G	1

In the above the characters are arranged in the descending order of their frequencies. The encoded data for the above data is as shown in the below table.

The decoding procedure is very simple. As the bits are are read, i.e when the first bit is read from the input, it is traversed from the beginning of the root by taking the left path or the right path in the binary tree depending on the the binary code 0 or 1 respectively. When the end of the tree is reached, characters would be decoded, and that character is placed on the output stream. So all the characters are decoded.

Character	Encoded Bits
E	11
I	10
T	0111
N	0110
L	0101
G	0100
O	0011
A	0010
M	00011
P	00010
U	00001
G	00000

From the above we can know that using Huffman's coding only 100 bits are required. If we don't use Huffman's coding then we need 232 bits (29 characters* 8 bits).

Optimized Huffmans Coding by grouping them in the set of 2:

Character	Frequency
EI	10
TN	6
LC	5
OA	4
MP	2
UG	2

The characters and their corresponding frequency. In this algorithm we group the letter into set of 2.
Here now for two letters we get one encoded bits so we must split the letter into two. We do it by taking the first set, in this case EI and split it into E and me. Now we check for the frequency of the letter, so the letter which gets highest frequency will be directly encoded with the obtained bit and the other character will be encoded with the addition of 0.

Character	Encoded Bits
EI	10
TN	100
LC	101
OA	110
MP	1110
UG	1111

Now that we have obtained the Encoding bits for the letters in the set of 2, we need to arrange them. As explained previously, in EI, E is having highest frequency so it will be encoded as 1 and I will be encoded with a prefix 0 i.e. 01. Here E is encoded as 1 while I is encoded as 01 and so on. So in this method we use only 82 bits to represent the above text.

Optimized Huffmans Coding by grouping them in the set of 3:

We have seen grouping of Huffman's coding in the set of two, now we will look at grouping in the set of three.
Consider the below TABLE of Huffman's coding grouped into set of three. In this algorithm we group the letter into set of 3 as show below.

Character	Frequency
EIT	13
NLC	8
OAM	5
PUG	3

To encode each bits we do it the same as that explained in the optimized Huffman's Coding for set of two. We find the letter with highest frequency and encode it. Here in this case, consider EIT where E has the highest frequency followed by I and then T. Now E is encoded directly with the obtained bits as 0, I is encoded with the prefix 0 i.e. the encoded bits is 00 and T is encoded with the prefix 1 to the obtained encoded bit i.e. 10 and so on. Here only 77 bits are sufficient to represent the given characters which shows an amazing compression rate.

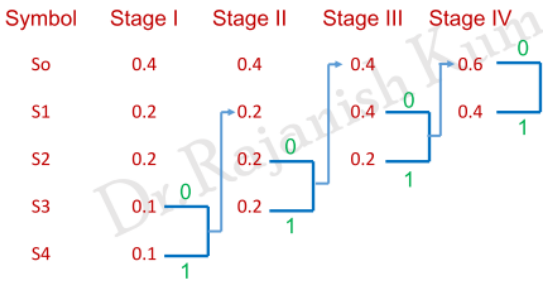
Character	Encoded Bits
EIT	0
NLC	10
OAM	110
PUG	111

When Huffman's coding is applied to the above TABLE considering each of the three letters as one we get the encoded bits

Algorithm:

The algorithm can be well understood after the above examples

- First, we arrange the letters in descending order of the frequency
- Second, we group the letters in the set of 2 or 3 or 4 etc. accordingly and then we tabulate them.
- Third, we perform Huffman's coding to each of the letter sets.
- Fourth, we obtain the encoded bits
- Next we see the frequency of the letter in each of the sets and arrange the letters in the specific set in descending order.
- Then we encode each letter. The letter with the highest frequency is encoded with the obtained bit directly. The following letter is encoded with the prefix 0, the following next letter with 1 and so on
- To decode we perform normal decoding procedure followed in Huffman's coding.



Symbol	Prob.	Code word
So	0.4	00
S1	0.2	10
S2	0.2	11
S3	0.1	010
S4	0.1	011

$$\bar{L} = 0.4(2) + 0.2(2) + 0.2(2) + 0.1(3) + 0.1(3)$$
$$= 2.2 \text{ binary symbols}$$

$$H(S) = 0.4 \log_2\left(\frac{1}{0.4}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right) + 0.1 \log_2\left(\frac{1}{0.1}\right) + 0.1 \log_2\left(\frac{1}{0.1}\right)$$
$$= 0.529 + 0.464 + 0.464 + 0.332 + 0.332$$
$$= 2.121 \text{ bits}$$

The average codeword length exceeds the entropy H(S) by only 0.0367.

The average codeword length does indeed satisfy:H(x) R < H(x)+1.

IV

Results

The results show that Optimized Huffman's coding can be implemented successfully. This approach helps research teams to make the results more efficient. This method is applied to text and the results are listed below. So the results are displayed categorically as:

Name	Encoded Bits	Compression Ratio
UTF-8	232 bits	0%
Huffman's coding	99 bits	57.33%
Optimized Huffman's Coding using set of 2	82 bits	63.37%
Optimized Huffman's Coding using set of 3 grouping	77 bits	76.30%

Compression ratio is calculated with respect to that of UTF8 coding.

V

Conclusion

The results obtained in this new method will be very helpful for many research works. It not only reduces the bits but also the tree because while decoding, the Huffman Tree is needed and in this method the Huffman tree is reduced to a great extent. Most of the researchers use huffmans coding in their algorithm and this optimized Hufmmans coding will help those researchers very much especially in signal processing applications and in various algorithms.

VI

ACKNOWLEDGMENT

This work was supported by Indian Institute of Information Technology Nagpur to promote research and innovation on huffman Coding Under the guidance of Dr. Rashmi Pandhare

VII

REFERENCES

- DAVID A. HUFFMAN, ASSOCIATE, IRE, “A Method for the Construction of Minimum-Redundancy Codes*”,1952
- Jas,A.Dept. of Electr. Comput. Eng., Univ. of Texas, Austin, TX, USA Ghosh-Dastidar, J. ; Mom-Eng Ng ; Touba, N.A., “ An efficient test vector compression scheme using selective Huffman”, Computer-Aided Design of Integrated Circuits and Systems, IEEE pp 797- 806, June 2003
- Gallager, R.G, “Variation of Theme by Huffman”, Information Theory 1978, pp.668–674.
- Karl Skretting, John Hakon and Sen Ole Aase, Hoskolen I Stavanger, Department of Electrical and Computer Engineering, “Improved Huffman Coding using Recursive Splitting”,1999, Research Gate