

Implementation and Performance Analysis of Huffman and Shannon-Fano Source Coding Using MATLAB

Ishwari Dhumal¹, Kanchan Borade², Anushka Boraste³, Priti Dhabale⁴, Dr. Sarika Jagtap⁵

¹Department of Electronics and Telecommunication Engineering, Maratha Vidya Prasarak Samaj's KBTCOE, Nashik, Maharashtra, India

Abstract— This paper presents a MATLAB implementation of both Huffman and Shannon-Fano coding. The symbols and their probabilities for both coding techniques are kept identical for both codes. The performances of the coding techniques are analyzed by finding the average code length, coding efficiency, redundancy, and entropy. The results obtained from MATLAB are validated by derivation. The results of the derivation are compared with the results obtained by MATLAB to check the correctness of the results. The two techniques, in general, provide a means of data compression; however, their approaches to data compression and the resulting structures are different.

Keywords—Huffman Coding, MATLAB, Shannon-Fano Coding, Source Coding, Data Compression.

I. INTRODUCTION

Data compression techniques reduce the size of data for storage and facilitate transfer over a communication channel. Compressed data is made up of less bits than the original data, and the compressed data should allow the storage or transmission of the original data [1]. The main idea behind data compression is representing data in a way that uses fewer bits. Huffman coding (lossless) determines the probability of the symbol and assigns the symbol with the least probability a longer code, and the symbol with the highest probability a shorter code. For a given set of symbols and their corresponding probabilities [2], the average code length and coding redundancy are determined using MATLAB for both Huffman coding and Shannon-Fano coding.

II. SHANNON-FANO CODING

A. Shannon-Fano Coding Principle

Shannon-Fano coding assigns a variable length binary code to the symbols of the message based on the probability that the symbol appears in the message. Code words are assigned to the more frequent symbols with a smaller number of bits and less frequent symbols with longer code words [3].

The method partitions the set of symbols based on their probability into two sub sets, and recursively repeats the process until each symbol is assigned a unique code word.

B. Shannon-Fano Coding Procedure

1. Place the source symbols in a list in descending order based on probability [4].
2. Group the symbols in such a way that the total probability of the symbols in each group is as near to each other as possible.
3. Assign the group containing the lower probability symbols the digit 0 and the other group containing the higher probability symbols the digit 1.
4. Repeat step 2 for the groups obtained in step 3.
5. Continue the process in step 4 until each symbol is assigned a binary digit code.
6. Find the average of the code lengths, the coding efficiency and the redundancy for the code constructed.

C. Shannon-Fano Code Construction

The symbols of the message in the given example are arranged in the order above in step 1. The symbols are then grouped in step 2. Shannon-Fano codes for the groups are shown in Table I with the corresponding code lengths.

TABLE I. SYMBOLS AND THEIR PROBABILITIES

Symbol	Probability
A	0.30
B	0.25
C	0.15
D	0.10
E	0.08
F	0.07
G	0.05

The branches of a binary tree are represented by the digits zero and one in the case of a Huffman coding scheme. Each element of the alphabet is represented by a code, which is determined by the path leading from the root node to that element.

The codes are determined by the construction of the binary tree. The codes are then evaluated to determine the average code word length.

TABLE II. SHANNON-FANO CODES AND CODE LENGTHS

Symbol	Probability	Shannon-Fano Code	Code Length
A	0.30	00	2
B	0.25	01	2
C	0.15	100	3
D	0.10	101	3
E	0.08	110	3
F	0.07	1110	4
G	0.05	1111	4

D. Theoretical Calculation of Shannon-Fano Coding

The Average code word length determines the entropy of a source, and coding scheme efficiency. The average code word length represents the highest possible coding scheme efficiency.

Average Code Length

$$L = \sum (P_i \times L_i)$$

$$L = (0.30 \times 2) + (0.25 \times 2) + (0.15 \times 3) + (0.10 \times 3) + (0.08 \times 3) + (0.07 \times 4) + (0.05 \times 4)$$

$$L = 0.60 + 0.50 + 0.45 + 0.30 + 0.24 + 0.28 + 0.20$$

$$\text{Average Code Length} = 2.57 \text{ bits/symbol}$$

Entropy

$$H = -\sum P_i \log_2(P_i)$$

$$H = 2.54 \text{ bits/symbol}$$

Coding Efficiency

$$\text{Efficiency} = (H/L) \times 100$$

$$\text{Efficiency} = (2.54/2.57) \times 100$$

$$\text{Efficiency} = 98.83\%$$

Redundancy

$$\text{Redundancy} = 100 - \text{Efficiency}$$

$$\text{Redundancy} = 100 - 98.83$$

$$\text{Redundancy} = 1.17\%$$

Theoretical Results of Shannon-Fano Coding

Parameter	Value
Entropy	2.54(bits/symbol)
Average Code Length	2.57(bits/symbol)
Coding Efficiency	98.83%
Redundancy	1.17%

E. MATLAB Implementation of Shannon-Fano Coding

The source code language MATLAB was used to construct a computer program to determine the Shannon-Fano coding scheme. The program was validated by comparing results to theoretical values.

F. MATLAB Output

```
Command Window
Entropy = 2.5400

Shannon-Fano Coding:
P = 0.30   Code = 00
P = 0.25   Code = 01
P = 0.15   Code = 100
P = 0.10   Code = 101
P = 0.08   Code = 110
P = 0.07   Code = 1110
P = 0.05   Code = 1111

Average Length = 2.5700
Efficiency = 98.83%
Redundancy = 1.17%
>>
```

III. HUFFMAN CODING

A. Huffman Coding Principle

Variable-length code words are created using Huffman coding by assigning code words of different lengths based on the probability of the symbol. Symbols with a greater probability are assigned a shorter length code word. Huffman coding creates a binary tree by combining the two symbols that have the lowest probability.

From the tree, a prefix code is determined for each of the source symbols. Prefix codes are importantly, codes in which no code word is a sub-code word of any other code word of the code [5].

B. Huffman Coding Procedure

Huffman coding is a technique that is used in data compression and is a common coding technique in communication. The procedures for determining Huffman codes for each of the symbols are described below.

1. The two symbols or nodes with the least probability are combined to form a new node [5],[6].
2. All the source symbols are then arranged in order from least to greatest probability.
3. The two symbols with the least probability are then combined to form a node, and the probability of the new node is determined by adding the probabilities of the two previous symbols.
4. The new node is then placed in the symbol list with the other nodes.
5. The procedure is repeated until there is only a single node.
6. Points 2 to 4 are repeated until all the available symbols are connected to a single root node.
7. Each of the branches of the Huffman tree gets a binary digit, 0 or 1.
8. The path from the root node to the symbol determines the code for that symbol.

Using the codewords created by the Huffman Coding process, we can calculate the average code word length, entropy, coding efficiency and the level of code word redundancy.

C. Huffman Code Generation

The Huffman codes are constructed using binary digits 0 and 1 corresponding to the two branching options of each node. The code word for a symbol is determined by the path from the root node to the symbol's leaf node. The Huffman codes and their corresponding code lengths are used in determination of the average code length and other performance evaluation parameters.

With the probability distribution shown above, the computed Huffman codes are given in Table III.

TABLE III. HUFFMAN CODES AND CODE LENGTHS

Symbol	Prob	I	II	III	IV	V	VI	Code	No.of bits/msg	
A	0.30	→0.30	→0.30	→0.30	0.43	0.57	0	1	00	2
B	0.25	→0.25	→0.25	0.27	0.30	0	0.43	1	01	2
C	0.15	→0.15	0.18	0.25	0	0.27	1		100	3
D	0.10	0.12	0.15	0	0.18	1			101	3
E	0.08	0.10	0	0.12	1				110	3
F	0.07	0	0.08	1					1110	4
G	0.05	1							1111	4

D. Theoretical Calculation of Huffman Coding

The average code length is computed using the Huffman codes and their lengths. Other performance metrics such as coding redundancy and coding efficiency are determined using the Huffman codes and codes lengths. The code lengths are obtained from the Huffman tree.

Average Code Length

$$L = \sum (P_i \times L_i)$$

$$L = (0.30 \times 2) + (0.25 \times 2) + (0.15 \times 3) + (0.10 \times 3) + (0.08 \times 3) + (0.07 \times 4) + (0.05 \times 4)$$

$$L = 0.60 + 0.50 + 0.45 + 0.30 + 0.24 + 0.28 + 0.20$$

$$\text{Average Code Length} = 2.57 \text{ bits/symbol}$$

Entropy

$$H = -\sum P_i \log_2(P_i)$$

$$H = -[(0.30 \log_2 0.30) + (0.25 \log_2 0.25) + (0.15 \log_2 0.15) + (0.10 \log_2 0.10) + (0.08 \log_2 0.08) + (0.07 \log_2 0.07) + (0.05 \log_2 0.05)]$$

$$\text{Entropy} = 2.54 \text{ bits/symbol}$$

Coding Efficiency

$$\text{Efficiency} = (H/L) \times 100$$

$$\text{Efficiency} = (2.54/2.57) \times 100$$

$$\text{Efficiency} = 98.83\%$$

Redundancy

$$\text{Redundancy} = 100 - \text{Efficiency}$$

$$\text{Redundancy} = 1.17\%$$

Theoretical Results of Shannon-Fano Coding

Parameter	Value
Entropy	2.54(bits/symbol)
Average Code Length	2.57(bits/symbol)
Coding Efficiency	98.83%
Redundancy	1.17%

E. MATLAB Implementation of Huffman Coding

The Huffman coding algorithm was executed in MATLAB for the given source probability distribution. The algorithm produces the Huffman code and calculates the code length, entropy, average code length, coding efficiency, and coding redundancy. The MATLAB implementation was performed to validate the Huffman coding procedure.

G. MATLAB Output

```

Command Window
Entropy = 2.5400

Huffman Coding:
P = 0.30   Code = 11
P = 0.25   Code = 01
P = 0.15   Code = 101
P = 0.10   Code = 001
P = 0.08   Code = 000
P = 0.07   Code = 1001
P = 0.05   Code = 1000

Average Length = 2.5700
Efficiency = 98.83%
Redundancy = 1.17%
>> |

```

IV. PERFORMANCE ANALYSIS

The Shannon-Fano and Huffman coding procedures were compared using average code length, entropy, coding efficiency, and coding redundancy. For uniformity, the probability distribution of source symbols was kept same for both coding procedures. The comparative analyses were performed using MATLAB and the results of coding procedures were obtained.

A. Performance Parameters

The given source has an entropy of about 2.54 bits per source symbol. For the probability distribution of the source, both Shannon-Fano and Huffman coding procedures give an average code length of about 2.57 bits. Hence, coding efficiency is about 98.83% and the redundancy is approximately 1.17%.

TABLE III. PERFORMANCE PARAMETERS OF SHANNON-FANO AND HUFFMAN CODING

Parameter	Shannon-Fano Coding	Huffman Coding
Entropy	2.54(bits/symbol)	2.54(bits/symbol)
Average Code Length	2.57(bits/symbol)	2.57(bits/symbol)
Coding Efficiency	98.83%	98.83%
Redundancy	1.17%	1.17%

B. Analysis of Coding Performance

The average code length for both Shannon-Fano and Huffman codes for the specified probability distribution is 2.57 bits/symbol. Thus, for this example, the two coding schemes have the same coding efficiency and redundancy [2],[7]. However, the code construction schemes for the two coding methods are different. Shannon-Fano coding constructs codes by successively subdividing the Symbols into subgroups of almost equal probability. On the other hand, Huffman coding constructs a code by successive unions of the two least probability nodes. Hence, the two coding schemes must yield codes of the same length for the same probability distribution.

C. Theoretical and MATLAB Verification

Entropy and average code length were calculated using MATLAB. The values obtained from the MATLAB calculations agreed with the values from the theory. Thus, the implementation of Shannon-Fano and Huffman coding was correct [8].

V. RESULTS AND DISCUSSION

The source symbol probability distribution function was used to implement both coding techniques. Theoretical calculations and results from MATLAB were used to evaluate the techniques.

The entropy of the source was calculated to be approximately 2.54 bits/symbol. The average code length of both coding techniques was less than or equal to 2.57 bits/symbol.

The coding efficiency of both methods was approximately 98.83% with approximately 1.17% redundancy. It was found that the codes represent the source data efficiently. It was observed that the average code length was approximately equal to the source data entropy, hence the codes made use of available source data information.

For the case of the selected probability distribution, it was found that the average code length for both Shannon-Fano and Huffman codes were the same, and, hence, the codes had the same coding efficiency and redundancy. The difference in the codes constructed by the two methods lies in the way probability is partitioned. Shannon-Fano codes employ probability partitioning in a recursive manner, and Huffman codes employ a bottom-up probability partitioning approach, using a binary merge of the least probable distribution cases.

Results obtained using MATLAB followed the trends predicted by theory. This proves the correct implementation of the codes and shows that both types of codes are efficient in lossless source coding.

VI. APPLICATIONS

Lossless coding is used in situations where data is to be stored or transmitted, and the stored or transmitted data must be the same as the original [9]. Variable length codes can be derived from the statistics of the data, using for example the Shannon-Fano or Huffman method. These codes have the following uses.

1. *Data Compression*: This refers to representing data in a compact form using code words where length of code words depends on the frequency of the symbol.
2. *Text Compression*: Compression of English text or other languages may be possible using a statistical coding method where the text can be exactly or approximately reconstructed.
3. *Communication*: Coding the data to be transmitted in a compact form increases the efficiency of the communication system.
4. *Media Compression*: Huffman coding is used for lossless compression of multimedia.
5. *Media Compression*: Huffman coding is used for lossless compression of media.

6. *Coding*: These algorithms are used to gain insight into the fundamental limits of lossless source coding.

VII. CONCLUSION

This work discussed the processes and evaluated the performances of Shannon-Fano coding and Huffman coding. The analyses were done using the same source symbol probability distribution. The coders were then developed and analyzed in MATLAB.

The entropy of the source was calculated to be around 2.54 bits/symbol. The average code length for both Shannon-Fano and Huffman codings was around 2.57 bits/symbol. The coding efficiency was 98.83% and coding redundancy was 1.17%.

Although the performances of the two methods are the same for this case, their procedures are completely different. For source coding, Shannon-Fano coding partitions the probability distribution into recursive smaller sets, while Huffman coding builds a binary tree of the probability distribution and substitution sets.

The two methods have various other applications in lossless source coding and data compression. The two methods have been implemented and analyzed to demonstrate their procedures [10].

REFERENCES

- [1] K. Sayood, Introduction to Data Compression, 5th ed., Cambridge, MA, USA: Morgan Kaufmann, 2017.
- [2] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, July 1948.
- [3] T. M. Cover and J. A. Thomas, Elements of Information Theory, 2nd ed., Hoboken, NJ, USA: John Wiley & Sons, 2006.
- [4] D. Salomon, Data Compression: The Complete Reference, 4th ed., London, U.K.: Springer, 2007.
- [5] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proceedings of the IRE, vol. 40, no. 9, pp. 1098–1101, September 1952.
- [6] D. J. C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge, U.K.: Cambridge University Press, 2003.
- [7] R. G. Gallager, Information Theory and Reliable Communication, New York, NY, USA: John Wiley & Sons, 1968.
- [8] MathWorks, "MATLAB Documentation," MathWorks, Natick, MA, USA.
- [9] I. H. Witten, R. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," Communications of the ACM, vol. 30, no. 6, pp. 520–540, June 1987.
- [10] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, vol. 27, no. 4, pp. 623–656, October 1948.