

Module 2 - Source Codes

* Introduction

- Source codes are used to compress or encode data in order to reduce its size for efficient storage and transmission
- The goal of source coding is to minimize the expected length of codeword while maintaining lossy or lossless compression
- A lossless compression ensures original data is reconstructed perfectly.
- A lossy compression allows some loss of information but still retains fidelity for practical purposes.

* Coding Parameters

- Various parameters are associated with source codes :-
- ① Code Length - The length of a codeword is the number of symbols in the sequence that represents it. Shorter codes provide better compression but potentially require complex decoding algorithms.

- ② **Code efficiency** :- The efficiency of code is a measure of how close it comes to achieving the theoretical lower bound on the avg. code length.
- ③ **Code Redundancy** - Redundancy in a code refers to the amount of additional information contained in the codeword length beyond requirement.
- Redundancy can be measured in terms of avg. codeword length and entropy.
- ④ **Code prefixness** - A prefix code is one in which no codeword is a prefix of another codeword. Prefix codes allow unambiguous decoding with no need for delimiters.
- ⑤ **Code Universality** - A universal code is one that can be used to compress any source with arbitrarily high probability of error free decoding.

* The Source code theorem.

→ Source coding Theorem / Shannon's coding Theorem is a fundamental result in information theory that establishes the theoretical limits of lossless data.

- It states that, for any source with H it is possible to construct a uniquely decodable code with an average codeword length close to H , and that no uniquely decodable code can achieve avg. codeword length less than H .
- So the source coding theorem sets a lower bound on the avg. length of codewords required to encode a source.
- This theorem has important applications in data compression, implying that no lossless compression algorithm can achieve rates greater than the entropy of source.

* Classification of codes

① Shannon - Fano Coding

- It is a loss-less data compression technique used to encode data taking up less space than before.
- The steps in the algorithm are:-
 - ① calculate probability of each symbol.

- ② Sort the symbols in decreasing probability.
- ③ More probable half is assigned a value $\{0, 1\}$ and less probable is opposite to more probable.
- ④ This is repeated recursively until all symbols are assigned a binary code.

Example ① \rightarrow Encode $[m] = [m_1, m_2, m_3, m_4, m_5]$

$$\{p\} = [0.30, 0.25, 0.15, 0.12, 0.08, 0.10]$$

m	$P(m)$	1 st	2 nd	3 rd	codeword	length
m_1	0.30	0	0	1	00	2
m_2	0.25	0	1	1	01	2
m_3	0.15	1	0	0	100	3
m_4	0.12	1	0	1	101	3
m_5	0.10	1	1	0	110	3
m_6	0.08	1	1	1	111	3

$$H = - \sum P_k \log_2 P_k$$

$$= - [0.30 \times \log_2 0.30 + 0.25 \log_2 (0.25) \\ + 0.15 \log_2 (0.15) + 0.12 \log_2 (0.12) \\ + 0.10 \log_2 (0.10) + 0.08 \log_2 (0.08)]$$

$$= + [0.521 + 0.5 + 0.41 + 0.367 + 0.332 \\ + 0.291]$$

$$= 2.421 \text{ bits}$$

$$L = \sum n \times P_k$$

$$2 \times 0.3 + 2 \times 0.25 + 3 \times 0.15 + 3 \times 0.12 \\ + 3 \times 0.10 + 3 \times 0.08$$

$$L = 2.45.$$

$$\therefore E \eta = H(S) / L$$

$$= \frac{0.988 - 2.42}{2.45} = 0.988 \\ = 98.8\%$$

Example ②

$$\left\{ \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8} \right\}$$

$$\{S_1, S_2, S_3, S_4\}$$

	1st	2nd	3rd	length
S ₁	1/2	0		1
S ₂	1/4	1	0	2
S ₃	1/8	1	1	3
S ₄	1/8	1	1	3

$$H(x) = - \left[\frac{1}{2} \log_2 \left(\frac{1}{2} \right) + \frac{1}{4} \log_2 \left(\frac{1}{4} \right) + \frac{1}{8} \log_2 \left(\frac{1}{8} \right) \right]$$

$$= \left\{ 0.5 + 0.5 + 0.375 + 0.375 \right\} \times 1.75$$

$\therefore = 1000$ bits

$$L = \sum P_k n_k R$$

$$= \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3$$

$\therefore = 1.75$ symbol.

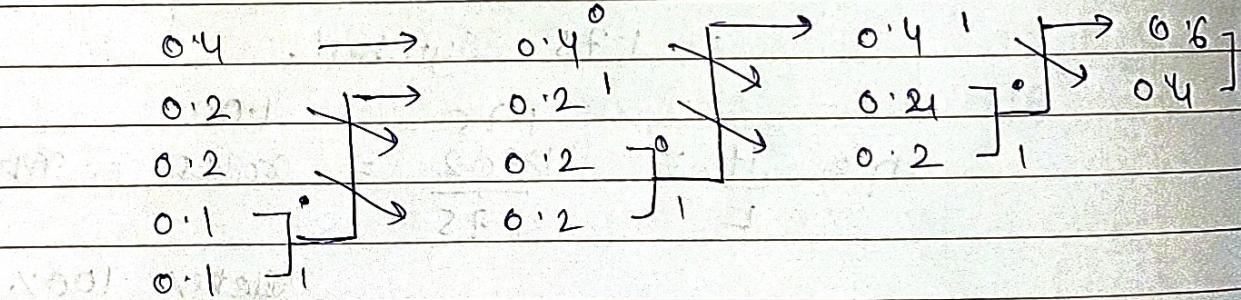
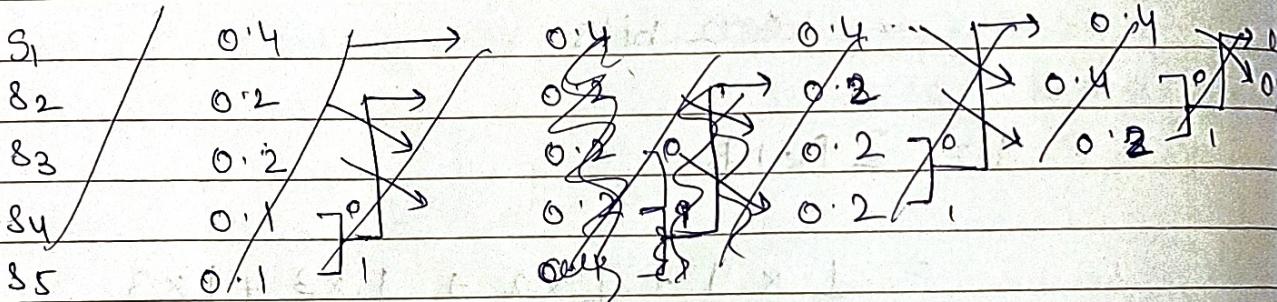
$$\eta = \frac{H}{L} = \frac{1.75}{1.00} = 0.995 = 99\%$$

accuracy: 100%.

* Huffman Coding.

- This algorithm works by first analyzing the data to be compressed and determining the frequency of occurrence of each symbol.
- Then a binary tree is constructed, with each leaf node representing a symbol and the frequency of that symbol.

Example ① $\rightarrow P = [0.4, 0.2, 0.2, 0.1, 0.1]$



		codeword	length
S ₁	0.4	00	2
S ₂	0.2	01	2
S ₃	0.2	11	2
S ₄	0.1	010	3
S ₅	0.1	011	3

$$H = -\sum P_n \log_2 P_n$$

$$= -[0.4 \log_2 (0.4) + 0.2 \log_2 (0.2) \times 2]$$

$$+ 0.1 \log_2 (0.1) \times 2]$$

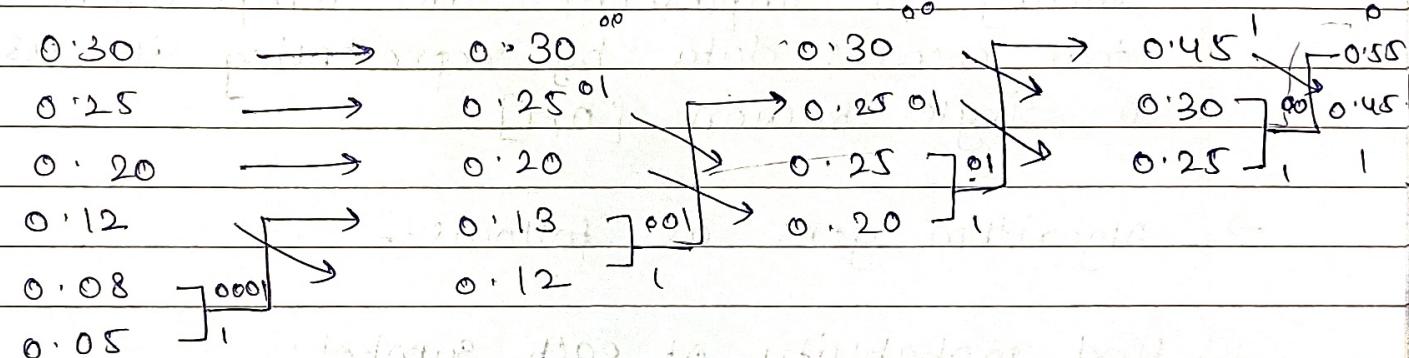
$$= -[0.528 + 0.928 + 0.664]$$

$$= 2.12 \text{ bits}$$

$$0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 \\ = 2.2 \text{ bits}$$

$$H = \frac{2.12}{2^2} = 0.963 \text{ bits}$$

Example ② - From the following data



codeword length

0.30	0000	2
0.25	0001	2
0.20	0011	2
0.12	0101	3
0.08	1000	4
0.05	1001	4

$$H = -[0.521 + 0.5 + 0.464 + 0.367 + 0.291 \\ + 0.216]$$

$$= 2.359 \text{ bits}$$

$$L = 0.3 \times 2 + 0.25 \times 2 + 0.2 \times 2 + 0.12 \times 3 + 0.08 \times 4 \\ + 0.05 \times 4.$$

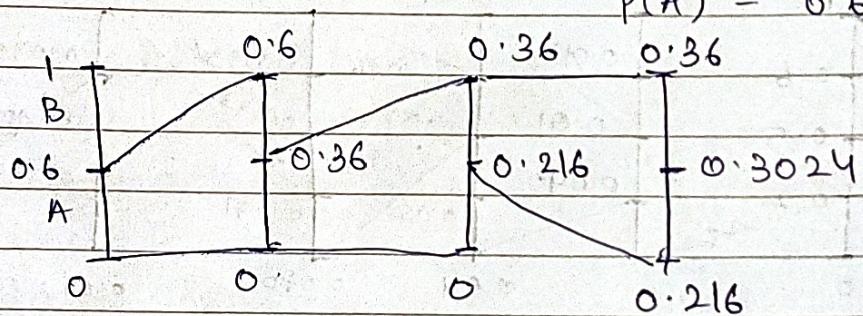
$$n = \frac{2.859}{2.38} = 99.11\gamma.$$

* Arithmetic Coding

- It is a technique used in information theory to implement data compression that encodes data by representing it as a single fraction $[0, 1]$.
- Algorithm goes as follows:-
 - ① Find probability of each symbol.
 - ② A range is assigned to each symbol.
 - ③ The entire message is then represented as a single fraction that falls within the range of the first symbol.
 - ④ The range keeps getting divided into sub-range that correspond to the probability distribution. This repeats till all the msg are encoded.

Example ①

$$P(A) = 0.6; P(B) = 0.4$$



$$R = LL + DX P$$

$$\begin{aligned} R &= 0 + 0.6 \times 0.6 \\ &= 0.36 \end{aligned}$$

$$\begin{aligned} R &= 0 + 0.36 \times 0.6 \\ &= 0.216 \end{aligned}$$

$$R = 0.216 + 0.144 (0.6)$$

$$\begin{aligned} &0.216 + 0.0864 \\ &= 0.3024 \end{aligned}$$

* LZW coding

→ LZW (Lempel - Ziv - Welch) coding is a lossless compression commonly used for GIF, TIFF, etc.

→ The algorithm works by replacing sequences of characters with codes. It starts by creating a dictionary that contains all possible single characters.

Example ① Encode ababba bc ababba
using LZW Coding

① Initializing the base dictionary.

index	entry
1	a
2	b
3	c

Dictionary

② Encode O/P

Index	Entry
1	a
2	b
3	c
4	ab
5	ba
6	abb
7	bab
8	bc
9	ca
10	aba
11	abba

1 1 2 4 8 2 3 4 6 1 1

Example ②

A A B A B B B A B A A B A B B B B A B B A B B

①

Index

Entry

1 A
2 B

②

Encode O/P

Index

Entry

1	A
2	B
3	AB
4	BA
5	ABB
6	BBA

Encode O/P

Index

Entry

-	1	A
-	2	B
1	3	AA
1	4	AB
2	5	BA
4	6	ABB
2	7	BB
5	8	BAB
5	9	BAA
4	10	ABA
3	11	AAB
8	12	BABB
7	13	BBA
13	14	BBAB

112425543 & 713

* Run Length Encoding

→ Run-length encoding is a simple and effective lossless data compression technique used to compress data with repeated values.

Simplest compression algorithm:

Example ① 0000011110010000101

{0, 5}, {1, 5}, {0, 2}, {1, 1}, {0, 4}, {1, 1}, {0, 1}, {1, 1}

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

$$\text{BITS} = \log_2 8 = 3 \text{ bits}$$

000

{0, 101}, {1, 101}, {0, 010}, {1, 001}, {0, 100}

{1, 001}, {0, 001}, {1, 001}

Final code -

0101101001010010100100100011001.

* Classification of Codes.

① Fixed Length codes

→ A fixed length code is one whose code word length is fixed.

② Variable - Length Codes

→ A variable length code is one whose code word length is not fixed.

③ Distinct Codes

→ A code word is distinct if each code word is distinguishable from each other.

④ Prefix - Free code

→ A code in which no code word can be formed by adding code symbols to another code word.

⑤ Uniquely decodable codes

→ A distinct code is uniquely decoded if the original source sequence can be reconstructed perfectly from encoded binary sequence.

* Classification of codes.

① Fixed Length codes

→ A fixed length code is one whose code word length is fixed.

② Variable - Length Codes

→ A variable length code is one whose code word length is not fixed. ~~ATM codes, rotatable codes~~

③ Distinct Codes

→ A code word is distinct if each code word is distinguishable from each other.

④ Prefix - Free code

→ A code in which no code word can be formed by adding code symbols to another code word.

⑤ Uniquely decodable codes

→ A distinct code is uniquely decoded if the original source sequence can be reconstructed perfectly from encoded binary sequence.

⑥ Instantaneous Codes

→ A uniquely decodable code is an instantaneous code if the end of any code word is recognisable without examining subsequent code symbols.

⑦ Optimal codes

→ A code is said to be optimal if it is instantaneous and has minimum average L for a given source

* Kraft Inequality

→ Let X be a DMS with alphabet $\{x_i\}$ ($i = 1, 2, \dots, m$)

$$\rightarrow K = \sum_{i=1}^m 2^{-n_i} \leq 1$$

Where n_i codeword length for x_i .

→ It assures us the existence of instantaneously decodable code with code word length that satisfy the inequality