DETAILED LECTURE NOTES

Unit-2

Source Coding Schemes for Data Compaction

Prefix Code

By considering the following code. {10, 00, 11, 110}

As per the chain shows that 11 is presents in beginning of the 110 or the 11 is the prefix of 110.

Example1 00 10 11 110

00 is immediately decodable bearuse there is no code coord starting from the 0

10 is immidiately codable

But, 11 cannot be decoded bearuse it can be 11 or 110, The Receiver has to waid for the next bit to receive to know whether 11 has to be transmitted or 110

Example 2. Now considering {0, 10, 110, 111}

As per the example no codeword is in the begining of any other codeword.

On it is the prefix free code. The given code is unique or it can be decoded instenously. Such type of code is called prefix code.

⟹ A prefix code is the code which is unique and instentenoly decoded.

⟹ A prefix code with codeword length n_1, n_2, n_L exists only and only if

$$\sum_{k=1}^{L} 2^{-n_k} \leq 1$$

Kraft inequality

when n is the number of bits in the k^{th} codeword.

for Example

VLC1 = 00 010 00 100 011 00 010 — 18 bits

letter	Codeword	letter	Codeword
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Passing Data: A BAD CAB

VLC2 = 0 1 0 01 00 0 1 — 9 bits

↓

letter	Codeword	letter	Codeword
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

So the Kraft inequality for both the codeword is

$$VLC_1 = 2^{-2} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-4} + 2^{-4}$$

$$= 0.25 + 0.125 + 0.125 + 0.125 + 0.125 + 0.125 + .0625 + 0.0625$$

$1 \leq 1$, satisfy Kraft inequality.

$VLC_2 = 2^{-1} + 2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} + 2^{-3} + 2^{-3}$

$= 0.5 + 0.5 + 0.25 + 0.25 + 0.25 + 0.25$

$+ 0.125 + 0.125$

$= 2.25 \neq 1$ does ~~not~~ satisfy Kraft inequality.

Thus the ~~reducing~~ the number is ~~not~~ the only goal for ~~increasing~~ the better information ~~but~~ codeword satisfaction, ~~redundancy~~ is already required

Average no ~~of~~ bit.

Average no ~~of~~ bit for a ~~set of~~ codeword is defined as

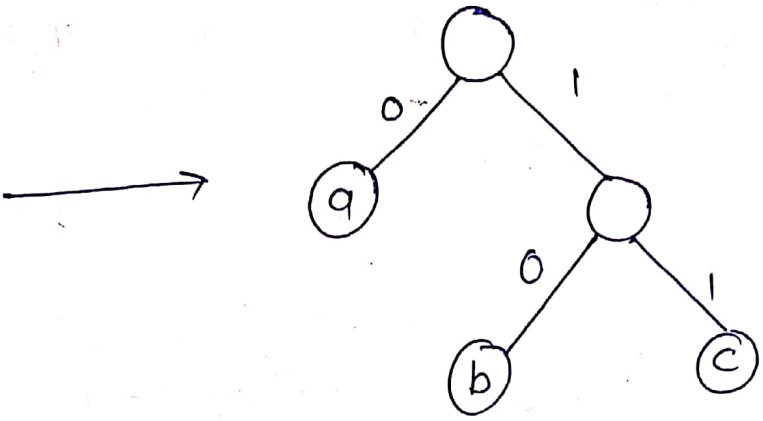
$$\overline{R} = \sum_{k=1}^{L} n(x_k) P(x_k)$$

n is the number ~~of~~ bits for a symbol x and P is the probablity ~~of~~ ~~that~~ symbol x.

It is also known as expected length ~~of~~ coding

Example check whether the code is prefix code or not

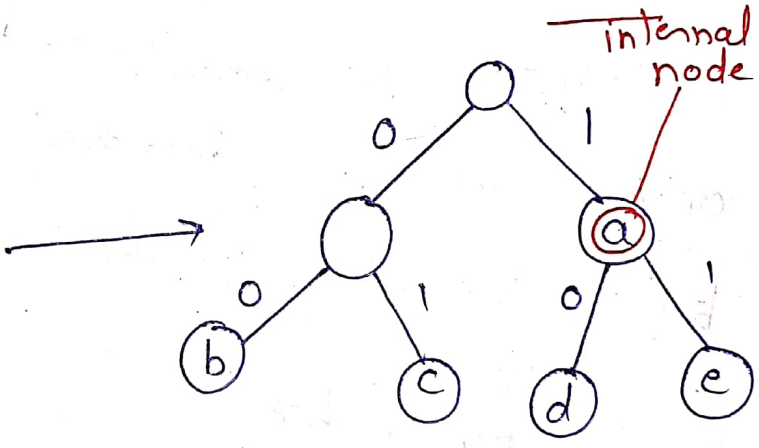
letter	code
a	0
b	10
c	11



from checking the prefix code. draw the binary tree if all the codewords one external code and in the leaves structure are known as prefix code.

Example 2

letters	code
a	1
b	0 0
c	0 1
d	1 0
e	1 1



in the showing binary tree (a) is in the middle part So this code word is not as the prefix code.

Huffman Coding

A variable length encoding was suggested by the Huffman based on the source symbol probabilities $P(x_i) = i = 1, 2 \cdots L$.

The algorithm is optimal in the sense that the symbol is provided minimum with the prefix condition is met.

Steps for the Huffman Coding

(i) Arrange the symbol in a decreasing order of their probablities.

2. Take the bottom two symbol and tie them together P_{n-1} $P_{n-1}+P_n$ P_n (0, 1)

3. Take the sum of probablity as new symbol. Again arrange the term in the decreasing order and add two numbers together. Each time the combination of two symbols we reduce the total no of symbols by one one. and labeling the two branch by 0 and 1.

4. Continue the procedure until only one probability is left. To complete the construction Huffman tree.

5. To find out the prefix code for any symbol follow the branches from the final node back to the symbol. While tracking back the route read out the labels on the branches.

Efficiency $\eta = \frac{H}{L \log_2 r}$

- for binary r will 2, (0,1)
- for ternary $r = 3$ (0,1,2)
- for Quaternary $r = 4$ (0,1,2,3)

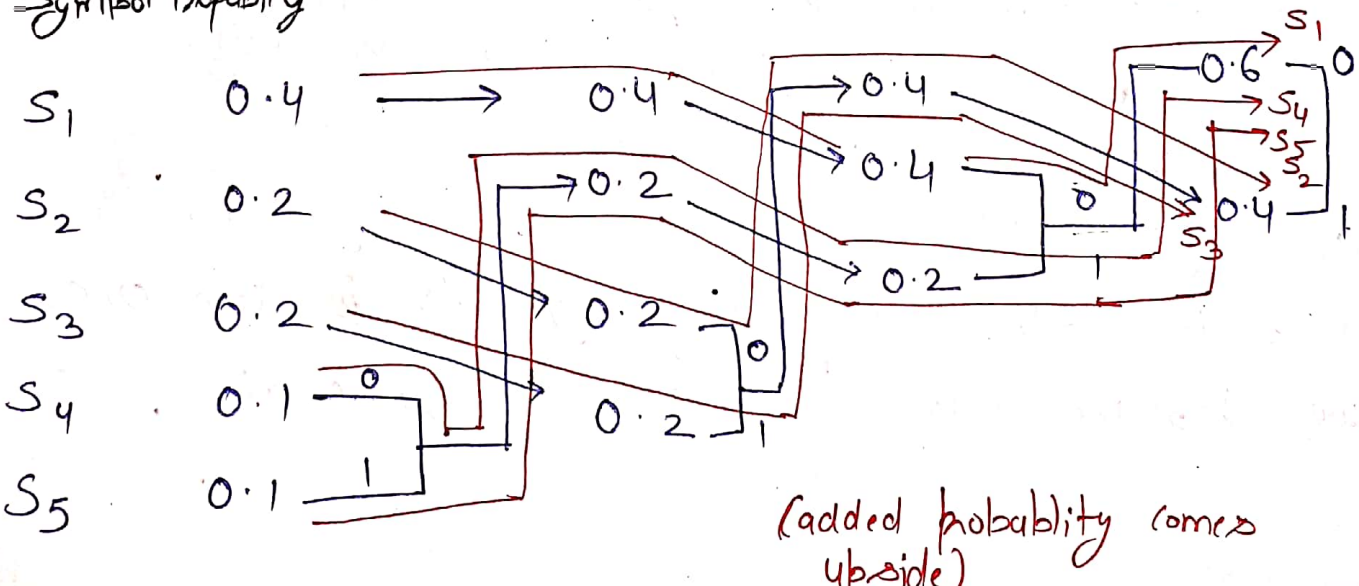
L = Avg Codeword $\sum_{i=1}^{n} P_i n_i$

$$H = \sum_{i=1}^{n} P_i \log_2 \frac{1}{P_i}$$

Example 1.

Abbtabet probablity = {0.4, 0.2, 0.2, 0.1, 0.1}
for symbol $\{S_1, S_2 \ldots S_5\}$ find Huffman code and also find efficiency and variance.

Symbol Probability



(added probability comes upside)

DETAILED LECTURE NOTES

Symbol	Codeword	length
S_1	00	2
S_2	10	2
S_3	11	2
S_4	010	3
S_5	011	3

Entropy $H = \sum P_i \log_2 \frac{1}{P_i}$

$0.4 \log_2\left(\frac{1}{0.4}\right) + 2 \times 0.2 \log_2\left(\frac{1}{0.2}\right)$

$+ 2 \times 0.1 \log_2\left(\frac{1}{0.1}\right)$

$= 2.1216$ bit/symbol

Average codeword length L

$L = \sum P_i n_i$

$= 2 \times 0.4 \times 2 (2 \times 0.2) \times 2 (3 \times 0.1)$

$= 2.2$ bl/symbol

Efficiency $\eta = \frac{H}{L \log_2 m}$ ($\because m = 2$ for binary)

$$\eta = \frac{2.1216}{2.2 \times \log_2 2} = 96.4\%$$

$$\boxed{\eta = 96.4\%}$$

Variance σ^2

$$\sigma^2 = \sum P_i (n_i - L)^2$$

$$= 0.4 (2 - 2.2)^2 + 2 \times 0.2 (2 - 2.2)^2 + 2 \times 0.1 (3 - 2.2)^2$$

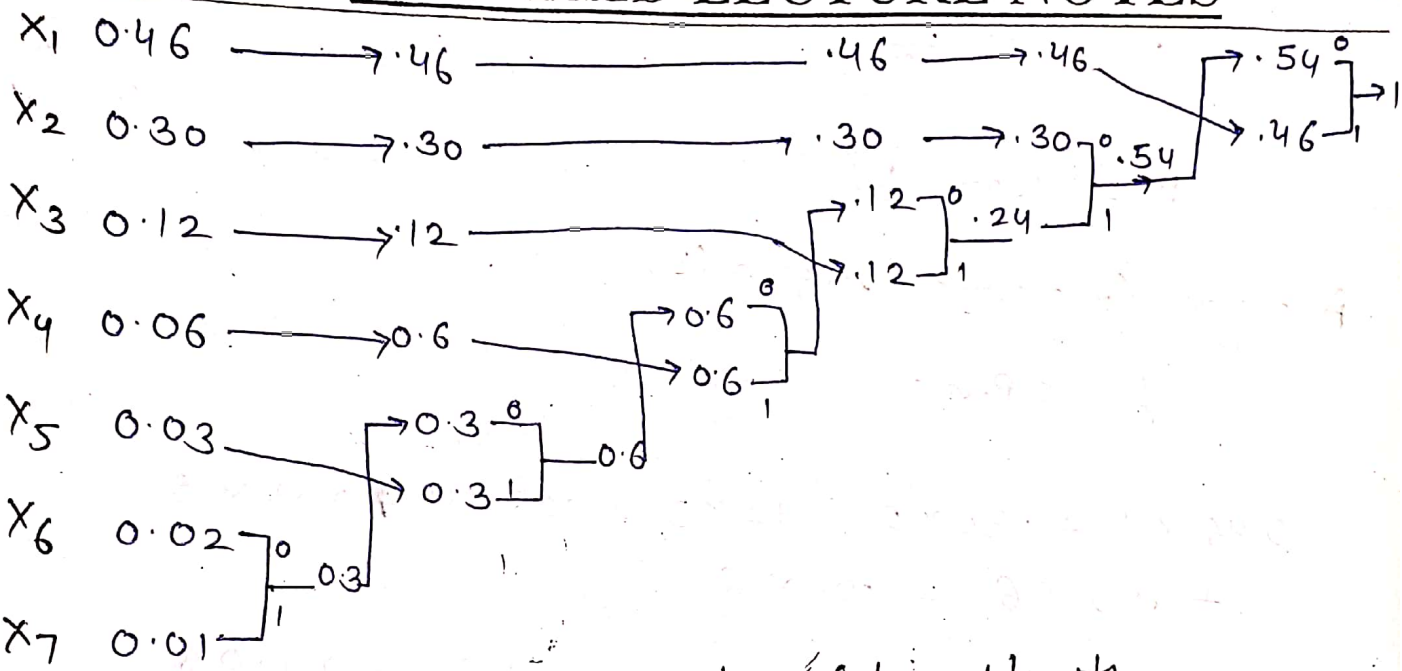
$$\boxed{\sigma^2 = 0.16}$$

It should be as low as possible so the coding ~~structure~~ very good.

Example 2

Consider a DMS with seven possible symbol x_i: $i = 1, 2, \ldots, 7$ and the corresponding probabilities $P(x_1) = 0.46$, $P(x_2) = 0.30$, $P(x_3) = 0.12$, $P(x_4) = 0.06$, $P(x_5) = 0.03$, $P(x_6) = 0.02$ and $P(x_7) = 0.01$

DETAILED LECTURE NOTES



Symbol	Probability	Codeword	Codeword length
x_1	0.46	1	1
x_2	0.30	00	2
x_3	0.12	011	3
x_4	0.06	0101	4
x_5	0.03	01001	5
x_6	0.02	010000	6
x_7	0.01	010001	7

Entropy $\sum P_k \log \frac{1}{P_k}$

$= .46 \log_2\left(\frac{1}{.46}\right) + .30 \log\left(\frac{1}{.30}\right) + .12 \log\left(\frac{1}{.12}\right)$

$+ .06 \log\left(\frac{1}{.06}\right) + .03 \log_2\left(\frac{1}{0.3}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right)$

$+ 0.1 \log_2\left(\frac{1}{0.1}\right)$

$H = 1.97$ bits

Average Code word length (L)

$L = \sum P_i n_i$

$= 0.46 \times 1 + .30 \times 2 + .12 \times 3 + .06 \times 4 + 0.03 \times 5$
$+ .02 \times 6 + .01 \times 6$

$= .46 + .60 + .36 + .24 + .15 + .12 + .06$

$= 1.99$ bit

Efficiency $\eta = \frac{H}{L \log_2 2}$

$\eta = \frac{1.97}{1.99 \times 1} = 98.9\%$

$\eta = 98.9\%$