جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الأمن السيبراني

Lecture: (6)

Fixed Length Code & variable Length Code

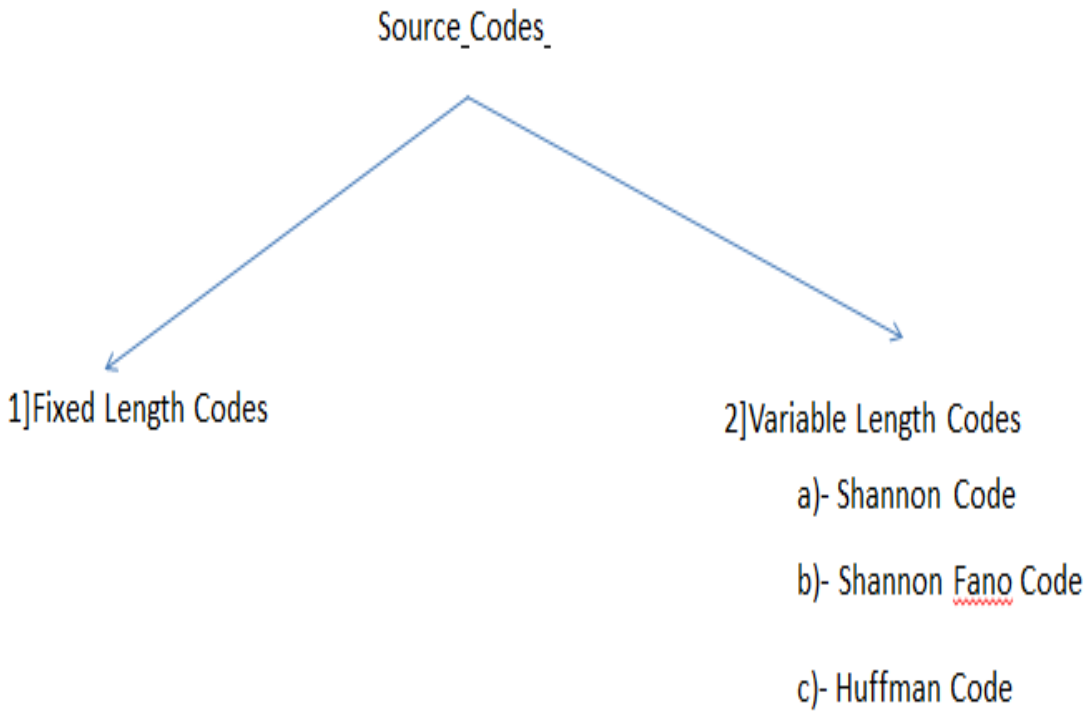
Subject: Coding Techniques
First Stage
Lecturer: Asst. Lecturer. Suha Alhussieny

Source Coding Techniques

1. Fixed Length Code

In fixed length coding technique all symbols assigned with equal length because the coding don't take the probability in account.

The benefit of the fixed length code is ease of applied (easy in coding and decoding)



This is used when the source produces almost equiprobable messages, $p(x_1) \approx p(x_2) \approx \ldots\ldots\ldots \approx p(x_n)$, then $L_{1=}L_{2=}\ldots\ldots=L_C$.

$L = li = \lceil log2\ M \rceil$ where M is the number of symbols.

If the alphabet X consists of the 7 symbols {a, b, c, d, e, f, g}, then the following fixed-length code of block length L = 3 could be used.

C(a) = 000

C(b) = 001

C(c) = 010

C(d) = 011

C(e) = 100

C(f) = 101

C(g) = 110.

The encoded output contains L bits per source symbol. For the above example the source sequence *bad*... would be encoded into 001000011... .

This method is non-probabilistic; it takes no account of whether some symbols occur more frequently than others, and it works robustly regardless of the symbol frequencies.

Example 1: A discrete source has an alphabet of five equiprobable letters. Evaluate the efficiency of a fixed length binary code.

Sol

$n=5 \quad \neq 2^r$ (binary D=2)

$L_c = \text{Int}[\log_D(n)]+1$
$= \text{int}[\log_2(5)]+1 = 3 \text{bits/letter}$

$H(X)=\log_2(n)$
$= \log_2(5)= 2.32 \text{ bits/letter}$

$\eta=H(X)/L_c= 2.32/3 = 77.3\ \%$

x_i	$p(x_i)$	Codeword C_i
x_1	0.2	000
x_2	0.2	001
x_3	0.2	010
x_4	0.2	011
x_5	0.2	100

Example 2: Let x = { x1,x2,……,x16 } where Pi = 1/16 for all i, fined $\xi source\ code$

Sol

H(x) = log2 M = log2 16 = 4 bits/symbol (because P1 = P2= ...=P16 = 1/M)

For fixed length code

$L = li = [log\ 2\ M]$

$L = li = [log\ 2\ 16] = 4$ bits/symbol.

Code Redundancy = L – H.

R = 4 - 4 = 0.

$$\therefore\ \xi_{source\ code} = \frac{H(x)}{L} * 100\% = \frac{4}{4} * 100\% = 100\%$$

source symbols	probability	codeword	code length
X_1	P_1	0000	4
X_2	P_2	0001	4
.	.	.	.
.	.	.	.
X_{16}	P_{16}	1111	4

Example 3: Let x = { x1,x2,…,x12 } where Pi = 1/12 for all i, fined $\xi source\ code$

Sol

H(x) = log2 M = log2 12 = 3.585 bit/symbol (because P1 = P2= ...=P12 = 1/M)

For fixed length code

$L = li = [log\ 2\ M]$

$L = li = [log\ 2\ 12] = [3.585] = 4\ bits/symbol$

Code Redundancy = L – H.

R = 4 – 3.585 = 0.415.

$$\therefore \ \xi_{source\ code} = \frac{H(x)}{L} * 100\% = \frac{3.585}{4} * 100\% = 89\ \%$$

Symbol	Probability	Code	l_i
X_1	1/12	0000	4
X_2	1/12	0001	4
.	.	.	.
.	.	.	.
X_{12}	1/12	1011	4

2- Variable length code

When the source symbols are not equally probable, a more efficient encoding method is to use variable-length code words. For example, a variable-length code for the alphabet X = {a, b, c} and its lengths might be given by

$$C(a)= 0 \quad l(a)=1$$

$$C(b)= 10 \quad l(b)=2$$

$$C(c)= 11 \quad l(c)=2$$

Example

Let U be a source taking values in {A, B, C, D} with probabilities ½, ¼, 1/8 and 1/8. 1,000 outputs of U are to be stored in the form of a file of binary digits, and one seeks to reduce the file to its smallest possible size.

First solution

There are **4 =2 ²** symbols to be encoded. Thus, each of them can be associated with a word of two binary digits as follows:

$A \rightarrow$ "00"
$B \rightarrow$ "01"
$C \rightarrow$ "10"
$D \rightarrow$ "11"

All the codewords having the number of bits, i.e. the same length, this code is said to be a **fixed length code**.

The size of the file is : **1000*2 =2,000** bits , **2 bits are used to represent one symbol.**

Second solution

The different symbols do not occur with the same probabilities. Therefore, we can think of a code which assigns shorter words to more frequent symbols as :

$A \rightarrow$ "1"
$B \rightarrow$ "01"
$C \rightarrow$ "000"
$D \rightarrow$ "001"

This code is said to be a **variable length code**, as the codewords do not have the same length.

$$1{,}000 \times \frac{1}{2} = 500 \text{ symbols of type "A"}$$

$$1{,}000 \times \frac{1}{4} = 250 \text{ symbols of type "B"}$$

$$1{,}000 \times \frac{1}{8} = 125 \text{ symbols of type "C"}$$

$$1{,}000 \times \frac{1}{8} = 125 \text{ symbols of type "D"}$$

Hence, the size of the file reduces to $500 \times 1 + 250 \times 2 + 125 \times 3 + 125 \times 3 = 1{,}750$ bits and is $\frac{2{,}000 - 1{,}750}{2{,}000} = 12.5\%$ smaller than it was in the previous solution, without loss of information (each symbol can be recovered reliably). The data have been compressed.

On average, $\frac{1{,}750}{1{,}000} = 1.75$ bit **are necessary to represent one symbol.**

If the symbol rate of U were 1,000 quaternary symbols per second, using the first code would result in a bit rate of 2,000 bits/sec. With the second code, the bit rate would reduce to 1,750 bits/sec.

- In a fixed-length code each codeword has the same length.
- In a variable-length code codewords may have different lengths.

Example

	a	b	c	d	e	f
Freq in '000s	45	13	12	16	9	5
fixed-len code	000	001	010	011	100	101
variable-len code	0	101	100	111	1101	1100

Note: since there are 6 characters, a fixed-length code must use at least 3 bits per codeword).

- The fixed-length code requires $300,000$ bits to store the file.
- The variable-length code requires only $(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1000 = 224,000$bits, saving a lot of space!

The ***advantage*** of a code where the message symbols are of variable length is that some time the code is more efficient in the sense that to represent the same information we can use ***fewer digits on the average***. To accomplish this we need to know something about the statistics of the message being sent. If every symbol is as likely as every other one, then the fixed length code are about as efficient as any code can be. But if some symbols are more probable than others, then we can take advantage of this feature to make the ***most frequent symbols correspond to the shorter encodings*** and the ***rare symbols correspond to the longer encodings***. This is exactly the idea behind variable length coding.

However, variable length code bring with them a fundamental ***problem***, at the receiving end, how do you recognize each symbol of the code? In, for example, a binary system how do you recognize the end of one code word and the beginning of the next ?

If the ***probabilities*** of the frequencies of occurrence of the individual symbols are ***sufficiently different*** , then ***variable – length encoding*** can be significantly ***more efficient*** than fixed ‑length encoding

$$P_i \uparrow \Longrightarrow l_i \downarrow$$

Example;

Symbol	Prob.	Code1
a1	0.49	1
a2	0.25	01
a3	0.25	001
a4	0.01	000

Table 1 : Variable-Size Codes.

In this case, the data has entropy :

$$H = -(0.49 \log_2 0.49 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.01 \log_2 0.01)$$
$$= -(-0.05 - 0.5 - 0.5 - 0.066) = 1.57 \text{ bits/symbol.}$$

(the number of bits per symbol) is :

$$L = \sum_{i=1}^{n} Pi\ li$$

$$= 1 \times 0.49 + 2 \times 0.25 + 3 \times 0.25 + 3 \times 0.01 = 1.77 \text{ bits/symbol.}$$

Which is very close to the minimum. The redundancy in this case is

$$R = L - H = 1.77 - 1.57 = 0.2 \text{ bits per symbol.}$$