Source Coding

We have seen that the source efficiency is defined as $H(X)/H(X)_{max}$.

Shannon Noiseless Source Coding theorem shows that the average number of binary symbols per source output can be made to approach the entropy of a source. In another word, the source efficiency can be made to approach unity, by means of source coding.

Figure 26

Sources with equal symbol probabilities, and/or statistically independent to each other, we can simply encode (map) each symbol into a codeword of block length *n*.

Example

Symbol '*A*' maps into codeword 00

Symbol '*B*' maps into codeword 01

Symbol '*C* maps into codeword 10

Symbol '*D*' maps into codeword 11 .

The length of each codeword is *fixed* at *n*.

n - block length of the code.

The code is called block code.

In some cases, sources with unequal symbol probabilities, and/or statistically non-independent symbols may have relatively poor source efficiencies by the above source encoding method (eg. source efficiency is much less than unity).

Example

The letter *E* in the English language occurs more frequently than the other letters.

If some symbols are more probable than others,then we take advantage of this to make the most frequent symbols associated with a codeword of shortest length. This type of encoding is called variable-length encoding. Four types of source encoding are described briefly below.

Shannon - Fano Encoding

From our previous discussion, we can find the average information content (entropy) of a source in bits per symbol. Here, we are dealing with the problem of how to convert an emitted symbol to a binary sequence. The process of symbol to binary sequence conversion is called source encoding and the device which does that, is called a source encoder.

At the receiving end, the received binary sequence (may be error free) is fed into a source decoder which performs the decoding operation and produces decoded symbols.

One method to implement the source encoder was proposed by Shannon and Fano. It is a variable-length encoding scheme and the algorithm is simply stated as below.

For M possible messages with probabilities p_1, p_2, ..., p_M and N symbols per message. A unique binary codeword c_i of length l_i for message i can be generated such that the average number of bits per symbol $\bar{H}_N$ is closer to G_N, where $G_N = -(1/N)\sum_i (p_i \log_2 p_i)$. In other words,

$$\bar{H}_N = (1/N) \sum_{i=1}^{M} l_i p_i \text{ -->} G_N . \tag{61}$$

Encoding steps:

1. Average the N messages in order of decreasing probability and let

$$F_i = \sum_{k=1}^{i-1} p_k \tag{62}$$

with

$$F_1 = 0 . \tag{63}$$

Example :

i	message	p_i	F_i
1	*AAA*	27/128	0
2	BBB	27/128	27/128 = $p_{i=1}$
3	*CAA*	9/128	54/128 = $p_{i=1} + p_{i=2}$

2. Calculate the number of bits i required for message i by the following equation.

$$-\log_2 p_i \le i < 1 - \log_2 p_i . \tag{64}$$

Example

i	p_i	l_i
1	27/128	3
2	27/128	3
3	9/128	4

3. Convert F_i into a binary fraction. A binary fraction satisfies the following equality

$$b_1 b_2 \ldots b_k = (b_1/2) + (b_2/2^2) + \ldots + (b_k/2^k) \tag{65}$$

Example 1 :

$27/128 = (0/2)+(0/2^2)+(1/2^3)+(1/2^4)+(1/2^6) +(1/2^7)$
$27/128 = (0/2)+(0/4)+(1/8)+(1/16)+(1/64) +(1/128)$

27/128 = . 0011011

Example 2 :

i	p_i	l_i	binary F_i
1	27/128	3	. 0000000
2	27/128	3	. 0011011
3	9/128	4	. 0110110

4. The codeword for message l_i is the truncated version of the binary fraction F_i and has length i.

Example :

l_i	binary F_i	Codeword
3	. 0000000	. 000
3	. 0011011	. 001
4	. 0110110	. 0110

High probability messages will have short codewords. Each message has a unique codeword and can be decoded uniquely.

The average number of bits per symbol (entropy) is bounded by

$$G_N \leq \bar{H}_N < G_N + (1/N) \qquad (66)$$

and can be derived from equation (64).

Finally, the encoder rate efficiency e is defined as

$$e = H / \bar{H}_N \qquad (67)$$

where $H = \sum_{i=1}^{n} P_i H_i$.

Decoding:

Consider a Markoff source encoder with N=2 symbols per message. The encoding operation is summarised in Table 1.

Table 1

Figure 27

The decoding is accomplished by taking the first $l_{i=1}$ -bit group and check for a matching word in Table 1. Finding none, we try the first $l_{i=3}$ -bit group and the process is repeated for higher order bit group.

Example

1. Take 11, the first 2-bit received group of digits.
2. Codewords 00 and 01 shown in Table 1 do not match the received word 11.
3. Take 1101, the first 4-bit received group of digits.
4. Pattern 1101 matches the codeword 1101 shown in Table 1 and decodes as symbol *BC*.

Huffman Encoding

Huffman encoding is an efficient source encoding method to remove redundancy. It is a variable-length encoding scheme.

The encoding method of a set of symbols may be determined as follows:

1. Arrange the source symbols in descending order of probability.
2. Create a new source with one less symbol by combining (adding) the two lowest probability symbols.
3. Repeat steps 1 and 2 until a single-symbol source is achieved.
4. Associate a ONE and a ZERO with each pair of probabilities so combined.
5. Encode each original source symbol into the binary sequence generated by the various combinations, with the first combination as the least significant digit in the sequence.

Example

Source symbols *A*, *B*, *C*, *D*, *E* and probabilities 0.4, 0.2, 0.2, 0.1, 0.1.

Figure 28

A encodes into 1

B encodes into 01

C encodes into 000

D encodes into 0010 (--.---.--- path)

E encodes into 0011

Figure 29 (Tree diagram)

Note that the Huffman's encoding rule is uniquely decodable; i.e., a stream of encoded symbols can be decoded without the need for synchronisation or framing pulses between the binary sequences representingeach symbol. The tree diagram confirms this property, since no encoded sequence is a prefix to another/any other valid sequence.

Eg. The encoded sequence for symbol *C* is not a prefix of any other valid sequence.

Calculations :

Source entropy $H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i$.

$H(X) = -0.4\log_2 0.4 - 0.2(2)\log_2 0.2 - 0.1(2)\log_2 0.1$

$H(X) = 2.12$ bits/symbol.

If $p_1 = p_2 = \ldots = p_m$,

$\max\{H(X)\} = -0.5(5)\log_2 0.5$

$\max\{H(X)\} = 2.322$ bits/symbol .

Source efficiency before encoding = $H(X) / H(X)_{max} = 0.913$.

The average number of binary digits used to encode each source symbol after encoding is

$$L = \sum_{i=2}^{m} p_i l_i \quad \text{bits/symbol} \qquad (68)$$

where l_i is the number of binary digits required to encode symbol i. For our example, $L = 2.2$.

The source efficiency η after encoding is

$$\eta = H(X)/L \; . \qquad (69)$$

1-η is defined as the source redundary.

M -- number of binary digits required to represent the source without any source coding.

L/*M* -- compression ratio *CR*

$$CR = L/M \; . \qquad (70)$$

For our example, *M*=3 and

$CR = 2.2/3 = 0.73$.

CR indicates the proportion by which the bandwidth required to transmit the source over a

binary channel can be reduced.

Summary

1. $L = \sum_{i=1} p_i l_i$

2. $\eta = H(X)/L$ — Source efficiency after coding

3. $H(X)/H(X)_{max}$ — Source efficiency before coding

4. $CR = L/M$ — Compression ratio

5. M — number of binary digits required to represent the source without coding

6. l_i — length of the encoded binary pattern for symbol i.

Figure 30

Tunstall Encoding

The encoding method of a set of symbols $\{s_1, s_2, ..., s_m\}$ may be determined as follows:

1. Arrange the m source symbols in descending order $\{s_1, s_2, ..., s_m\}$ of probability; i.e.,

   $$p_1 \geq p_2 \geq \ldots \geq p_m.$$

2. Create a new source with $2m$-1 symbols by splitting the highest probability symbol into m symbols with probabilities $p_1{}^2, p_1p_2, p_1p_3, \ldots, p_1p_m$.

   Let the corresponding symbols obtained as $s_1s_1, s_1s_2, s_1s_3, \ldots, s_1s_m$.

3. Repeat step 2 until the number of symbols in the new source is a power of b, if the encoding is to be into binary digits b=2.

4. Use each binary sequence to encode those sequences emitted by the original source which correspond to the labels of the new source symbols.

Example 1

A binary source with $p_1 = 0.75$ and $p_1 2 = 0.25$. m= 2, b=2 .

$$H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

$H(X) = 0.811$ bit/symbol .

$H(X)_{max}$ = -0.5 $\log_2$ 0.5 - 0.5 $\log_2$ 0.5

$H(X)_{max}$ = 1 bit/symbol .

Source efficiency before encoding = $H(X) / H(X)_{max}$ = 81.1% .

Figure 31

Total probability = $p_1 + p_2 = 1$

$p_1 = 0.75$

$p_2 = 0.25$

$p_1^2 = 0.5625$

$(p_1^2)p_1 = 0.422$

$p_2 p_1 = 0.1875$

$(p_2 p_1)p_1 = 0.141$

Step 1 : $p_1 \geq p_2$

Step 2 : a new source with symbols

$\{s_1 s_1, s_1 s_2, s_2\}$ with probabilities
$\{p_1^2, p_1 p_2, p_2\}$

Step 3 : a new source with symbols

$\{s_1 s_1 s_1, s_1 s_2 s_1, s_1 s_2, s_2\}$ with probabilities
$\{(p_1)^2 p_1, (p_2 p_1)p_1, p_2 p_1, p_2\}$

Step 4 : Assign the followings

symbol	maps	binary sequence
$s_1 s_1 s_1$		11
$s_1 s_2 s_1$		10
$s_1 s_2$		01
s_2		00

Step 5 : Encode the followings

source O/P	encodes	binary sequence
1		00
01		01
001		10
000		11

Example 2

If the original source emitts the sequence

01 000 1 001 000 = 12 symbols

then the encoded sequence is

01 01 00 10 11 = 10 symbols .

Figure 32

On average,

$$M = \sum_{i=1}^{m'} p_i' l_i' \quad . \tag{71}$$

M - the number of binary digits required to represent the source without source coding

l_i' -- length of the i-th symbol without source coding

p_i' -- probability associated with symbol i

m' -- total number of symbols

L -- the average number of binary digits used to encode each source symbol

For the example 1, with symbols

$\{s_1s_1s_1, s_1s_2s_1, s_1s_2, s_2\}$

$$M = p_2(1)+p_2p_1(2)+(p_2p_1)p_1(3)+(p_1^2)p_1(3)$$

$$= 2.314 \ .$$

The compression ratio = L / M = 0.866 .

The source efficiency after encoding

$$= MH(X) / L$$

$$= H(X) / CR \qquad (72)$$

$$= 0.938 .$$

Run-Length Coding

Run-length coding technique is particularly suitable for binary sources where one of the symbols (the ONE, say) occurs very much less often than the other, so that there are long runs of successive ZEROS (e.g., a scanned and digitised line drawing). In this case, it is more efficient to encode by counting the number of consecutive ZEROS between ONES.

Example

Figure 33

If a binary source generates a sequence of 20 digits

001, 000000, 1, 1, 0000001,

This sequence is encoded into a 15-digit sequence

010, 111, 000, 000, 110 .

The compression ratio in this case is

15/20 = 0.75 .

Again, run-length coding is still a variable-length encoding scheme. In this case,

Input block length -- varies

Output block length -- fixed

Reference

[1] Shanmugam, K. S., 'Digital and Analog Communication Systems', John Wiley & Sons, 1979.

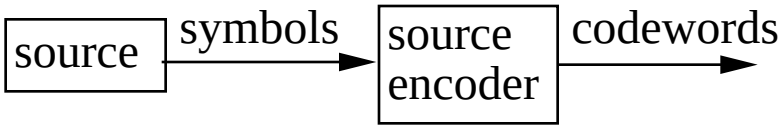

Figure 26

i	symbols	p_i	l_i	codeword
1	AA	9/32	2	00
2	BB	9/32	2	01
3	AC	3/32	4	1001
4	CB	3/32	4	1010
5	BC	3/32	4	1100
6	CA	3/32	4	1101
7	CC	2/32	4	1111

$\bar{H}_N$ = 1.44 bit/symbol

Table 1

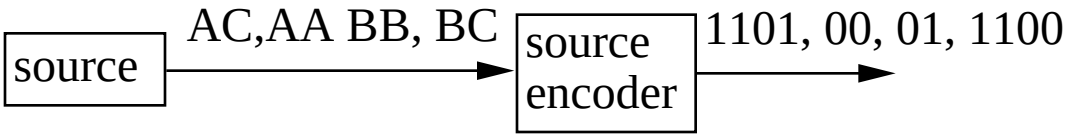

Figure 27

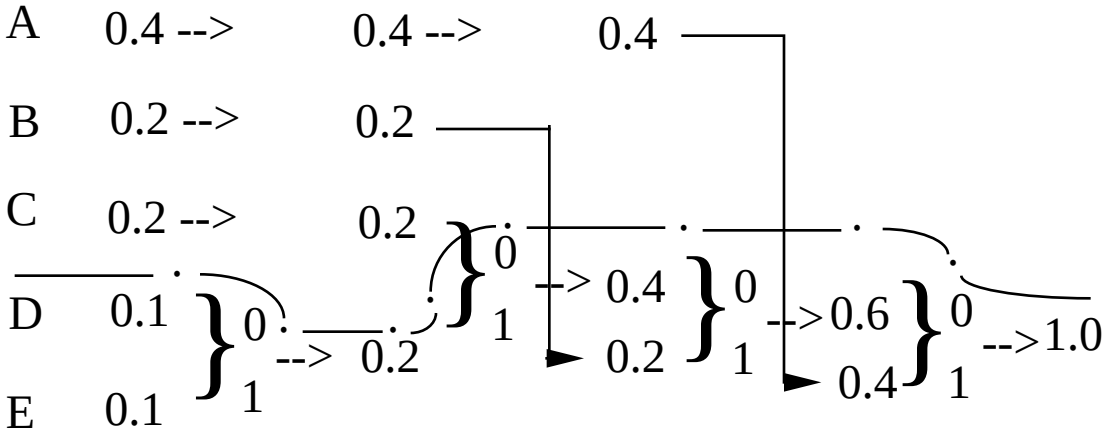

Figure 28

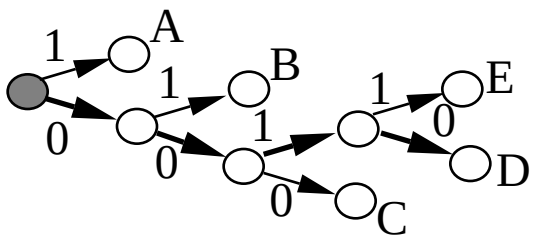

Figure 29

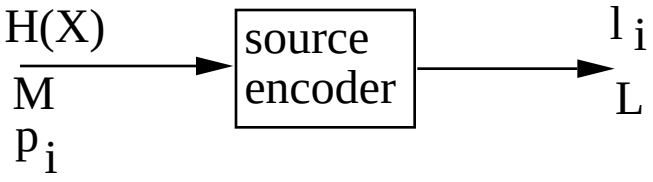

Figure 30

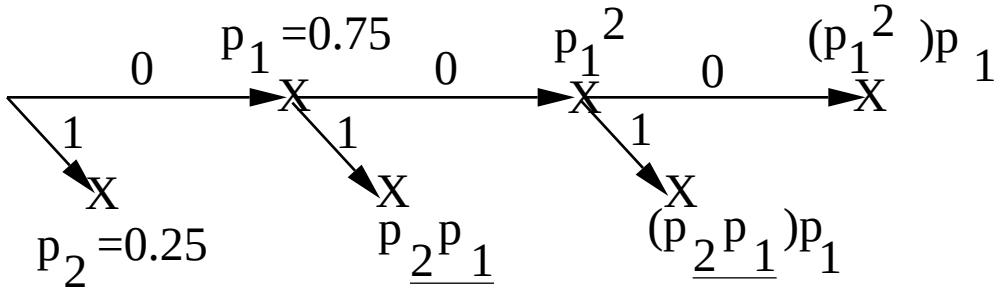

Figure 31

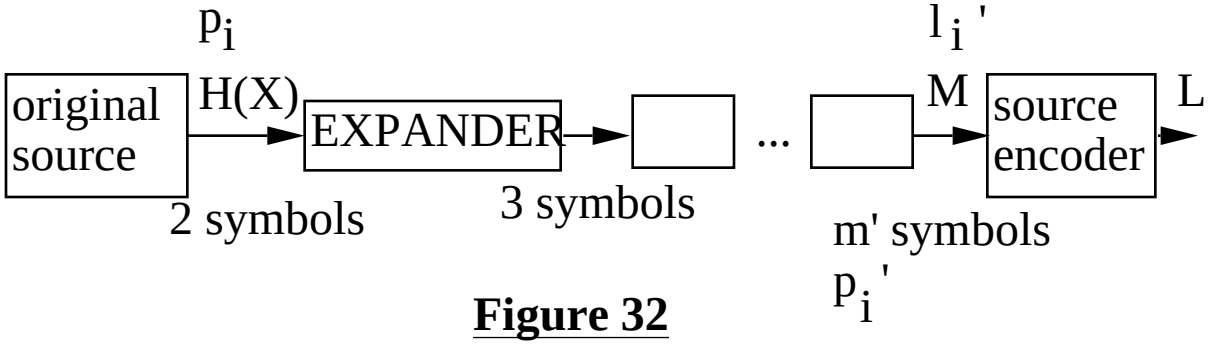

Figure 32

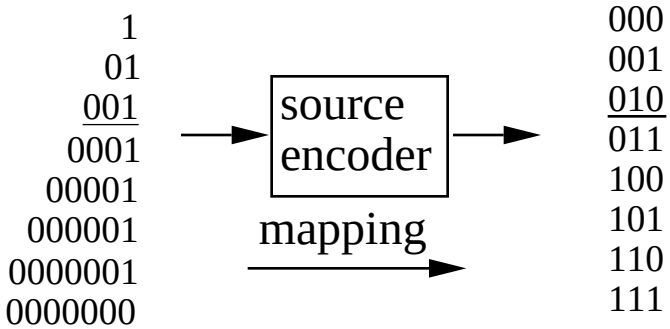

Figure 33