Huffman coding

Optimal statistical coding

- ***Definitions***:
  - S: discrete and simple source of messages m_i with probability law $p = (p_1, \ldots\ldots\ldots\ldots\ldots, p_N)$ (homogeneous source)
  - Coding of an alphabet $A = \{ a_1, a_2, \ldots\ldots, a_q \}$
  - Entropy of the source H(S) and average length of code-words E(n)
- ***MacMillan's theorem***:
  - There exists at least one irreducible inverting code that matches:

$$H / \log_2 q \quad E(n) < (H / \log_2 q) + 1$$

Equality if p_i of the form: $p_i = q^{-ni}$ (if $q = 2$ => $n_i = - \log_2 p_i$)

- ***Shannon's theorem*** (1st theorem on noiseless coding)

$$H / \log_2 q \quad E(n) < (H / \log_2 q) +$$

In the previous resource, "Defining coding and properties" we saw that the objectives of coding are mainly to transcribe information and to reduce the quantity of symbols necessary to represent information. By optimizing the coding, we attempt to reduce the quantity of symbols as much as possible.

Let's consider a simple source of homogeneous information $S = \{m_1, m_2, \ldots, m_N\}$ armed with a probability law $p = \{ p_1, p_2, \ldots, p_N \}$ où $p_i = Pr\{m = m_i\}$.
If M_i is the code-word that corresponds to the message m_i, we call $n_i = n(M_i)$ the number of characters that belong to the alphabet A (Card(A) = q) needed for the coding of m_i, n_i is thus the length of the code-word M_i.

The average length of the code-words is then: $E(n) = \sum_{i=1}^{N} p_i n_i$.

The average uncertainty of a source is the entropy H. The average uncertainty per character of the alphabet A is equal to $\frac{H}{E(n)}$, so we get: $\frac{H}{E(n)} \leq \log_2 q$ because alphabet A contains « q » characters. From this inequality, we deduce that $E(n) \geq \frac{H}{\log_2 q}$.

To optimize coding, we want to reduce E(n), the average length of the code-words. This average length cannot be lower than $\frac{H}{\log_2 q}$. Nevertheless, two theories show that it is possible to obtain equality $E(n) = \frac{H}{\log_2 q}$ and an optimal code:

Mac Millan's theorem:

An information source S with entropy H coded by an inverting way with an alphabet counting q characters is such that: $E(n) \geq \frac{H}{\log_2 q}$ and there exists at least one irreducible code of a given law such as $E(n) \leq \frac{H}{\log_2 q} + 1$. Equality is reached if $p_i = q^{-ni}$ (i.e. $n_i = - \log_q p_i$) and we then have an optimal coding.

Note:
In the particular case of a binary alphabet {0, 1}, we have q = 2. If the relationship $n_i = -\log_2 p_i$ is true, we then have E(n) = H: the entropy is the bottom limit of the set of code-word average lengths and this lower limit is reached.

Shannon's theorem of noiseless coding:

Any homogeneous source of information is such as there exists an irreducible coding for which the average length of code-words is as close as we want to the lower limit $\frac{H}{\log_2 q}$.
The demonstration of this theorem uses irreducible coding in blocks (block coding assigns a code-word to each block of « k » messages of S, consecutive or not).

Optimal statistical coding

- ***Fano - Shannon coding***

- ***Arithmetic coding*** ***(block encoding, interval type encoding)***
  possibilities of on line adaptation

- ***Huffman coding***
  3 basic principles:
  - if $p_i < p_j \Rightarrow n_i \quad n_j$
  - the 2 unlikeliest codes have the same length
  - the 2 unlikeliest codes (of max length) have the same prefix of length $n_{max}-1$

The presentation above shows three types of codes that are close to optimal code:

Shannon-Fano coding:

It tries to approach as much as possible the most compact irreducible code, but the probability p_i is not usually equal to 2^{-ni}, so the coding can only be close to optimal coding.
The probabilities p_i associated with the messages m_i are arranged by decreasing order then we fix n_i so that: $n_i \geq \log_2 \frac{1}{p_i} \geq n_i - 1$. Finally, we choose each code-word M_i of length n_i so that none of the previously chosen code-words forms a prefix (it avoids decoding ambiguity cf.: *"classification of the codes"*).

Arithmetic coding:

The code is associated with a sequence of messages from the source and not with each message. Unlike Huffman coding, which must have an integer length of bits per message and which does not always allow an optimal compression, arithmetic coding lets you code a message on a non-integer number of bits: this is the most effective method, but it is also the slowest. The different aspects of this coding are more fully developed in the resource: "*Statistical coding: arithmetic coding*".

Huffman coding:

Huffman coding is the optimal irreducible code. It is based on three principles:

- if $p_j > p_i$ then $n_i \leq n_j$,
- the two most unlikely words have equal lengths,
- the latters are written with the same n_{max}-1 first characters.

By using this procedure iteratively, we build the code-words M_i of the messages m_i.

- Example:

Let be a source $S = \{m_1, m_2, \ldots, m_8\}$ with a probability law: $p_1 = 0.4$; $p_2 = 0.18$; $p_3 = p_4 = 0.1$; $p_5 = 0.07$; $p_6 = 0.06$; $p_7 = 0.05$; $p_8 = 0.04$.

We place these probabilities in decreasing order in the column $p_i^{(0)}$ of the table below. We can see that in the column $p_i^{(0)}$ the probabilities of the messages m_7 and m_8 are the smallest, we add them and reorder the probabilities, still in decreasing order, to create the column $p_i^{(1)}$:

Messages	$p_i^{(0)}$	$p_i^{(1)}$
m_1	0,4	0,4
m_2	0,18	0,18
m_3	0,1	0,1
m_4	0,1	0,1
m_5	0,07	0,09
m_6	0,06	0,07
m_7	0,05	0,06
m_8	0,04	

Generally, we add the two smallest probabilities in the column $p_i^{(k)}$, then we reorder the probabilities in decreasing order to obtain the column $p_i^{(k+1)}$. Finally we get the following table:

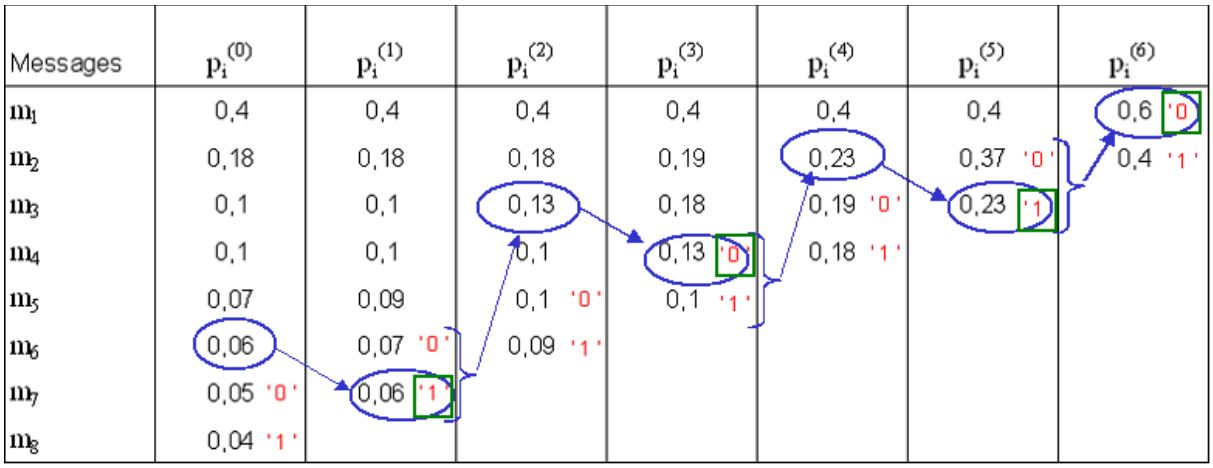
Messages	$p_i^{(0)}$	$p_i^{(1)}$	$p_i^{(2)}$	$p_i^{(3)}$	$p_i^{(4)}$	$p_i^{(5)}$	$p_i^{(6)}$
m_1	0,4	0,4	0,4	0,4	0,4	0,4	0,6
m_2	0,18	0,18	0,18	0,19	0,23	0,37	0,4
m_3	0,1	0,1	0,13	0,18	0,19	0,23	
m_4	0,1	0,1	0,1	0,13	0,18		
m_5	0,07	0,09	0,1	0,1			
m_6	0,06	0,07	0,09				
m_7	0,05	0,06					
m_8	0,04						

We assign the bits ‘0’ and ‘1’ to the last two elements of each column:

Messages	$p_i^{(0)}$	$p_i^{(1)}$	$p_i^{(2)}$	$p_i^{(3)}$	$p_i^{(4)}$	$p_i^{(5)}$	$p_i^{(6)}$
m_1	0,4	0,4	0,4	0,4	0,4	0,4	0,6 '0'
m_2	0,18	0,18	0,18	0,19	0,23	0,37 '0'	0,4 '1'
m_3	0,1	0,1	0,13	0,18	0,19 '0'	0,23 '1'	
m_4	0,1	0,1	0,1	0,13 '0'	0,18 '1'		
m_5	0,07	0,09	0,1 '0'	0,1 '1'			
m_6	0,06	0,07 '0'	0,09 '1'				
m_7	0,05 '0'	0,06 '1'					
m_8	0,04 '1'						

For each message m_i, we go through the table from left to right and in each column we can see the associated probability $p_i^{(k)}$ (blue path on the illustration below).
The code-word M_i is then obtained by starting from the last column on the right and moving back to the first column on the left, by selecting the bits associated with the probabilities $p_i^{(k)}$ of the message m_i (green rectangles on the illustration below).

For example, we want to determine the code-word M_6 of the message m_6. We detect all the probabilities $p_6^{(k)}$:



The code-word M_6 is thus obtained by simply reading from right to left the bits contained in the green rectangles: '0' – '1' – '0' – '1'. By following the same procedure for each message, we obtain:

Messages	Huffman codes
m_1	1
m_2	0 0 1
m_3	0 1 1
m_4	0 0 0 0
m_5	0 1 0 0
m_6	0 1 0 1
m_7	0 0 0 1 0
m_8	0 0 0 1 1

The average length of the code-words is equal to:

$$E(n) = \sum_{i=1}^{8} p_i n_i = 0.4\times1 + 0.18\times3 + 0.1\times3 + 0.1\times4 + 0.07\times4 + 0.06\times4 + 0.05\times5 + 0.04\times5$$

$$E(n) = 2.61$$

We can compare this size with the entropy H of the source:

$$H = -\sum_{i=1}^{8} p_i \log_2 p_i = 2.552$$

The efficiency η of the Huffman coding for this example is thus $\frac{2.553}{2.61} = 97.8\ \%$. For comparison purposes, 3 bits are needed to code 8 different messages with a natural binary ($2^3 = 8$). For this example, the efficiency of the natural binary coding is only $\frac{2.553}{3} = 85\ \%$.