CDI15

3. *Probability coding*

303 SXD

3.1. Basic notions about source coding

Generalities

Let $\mathcal{A} = \{a_1, \dots, a_n\}$ *source alphabet*, so that the *source* emits a stream of *symbols* from $\mathcal{A}$. At this stage we assume that the source is *memoryless*, which means that if X_i denotes the i-th symbol (so X_i is a random variable with values in $\mathcal{A}$), then

$$p_j = P(X_i = a_j)$$

is independent of i, and hence also independent of all other symbols in the stream.

Let $\mathcal{C}$ be a *coding* alphabet of r symbols, as for example the binary alphabet ($r = 2$). An *encoding* of $\mathcal{A}$ is a map

$$f: \mathcal{A} \to \mathcal{C}^*, a_i \mapsto f(a_i).$$

The elements of the list $C_f = \{f(a_1), \dots, f(a_n)\}$ will be called the *code-words* of the encoding f.

The f-encoding of a *source message* $M = a_{i_1} a_{i_2} \cdots a_{i_k} \in \mathcal{A}^*$ is the string $f(M) \in \mathcal{C}^*$(*coded message*) defined by

$$f(M) = f(a_{i_1}) f(a_{i_2}) \cdots f(a_{i_k}) .$$

Thus we can regard f as a map $f\colon \mathcal{A}^* \to \mathcal{C}^*$.

The encoding f is said to be *uniquely decodable/decipherable* (ud) if this map is 1-to-1, which means that any string from $\mathcal{C}^*$ is at most the image of one string from $\mathcal{A}^*$.

Example. The encoding $f\colon \{x, y, z\} \to \{0,1\}^*$ such that

$$f(x) = 0,\ f(y) = 10,\ f(z) = 010$$

is not uniquely decodable, as 010 can be decoded as z and as xy.

If no code word is a prefix of any other code word, we say that the encoding (or the code) is *prefix* (or *instantaneous*). Such codes are uniquely decodable.**N1**

Example. The encoding $x \to 0$, $y \to 01$ in not prefix, but it is ud.**N2**

Henceforth, encoding will mean uniquely decodable encoding. As we have seen, prefix encodings (and hence in particular *block encodings*, whose words have all the same length) are encodings in this sense.

The *code* of and encoding $f: \mathcal{A} \to \mathcal{C}^*$ is the set C_f of codewords $\{c_1, \dots, c_n\}$, $c_j = f(a_j)$.

Remark. Note that the ud property can be stated by saying that if

$$c_{i_1} \cdots c_{i_k} = c_{i'_1} \cdots c_{i'_{k'}} \ ,$$

then $k' = k$ and $i'_s = i_s$ for $s = 1, \dots, k$. In other words, that the expression of an element of $\mathcal{C}^*$ as concatenation of elements of C is unique.

Remark. The question of whether a subset C of $\mathcal{C}^*$ has this property can be decided by the Sardinas-Patterson algorithm (1953): It takes C as input and outputs either that C has the property or an element of $\mathcal{C}^*$ that has two different expressions as concatenations of elements of C.

3.2. Construction of prefix codes

Suppose $f: \mathcal{A} \to \mathcal{C}^*$ is an encoding. Let $C_j \subseteq \mathcal{C}^j$, $j = 1, \dots, l$, be the set of code words of length j, where l denotes the maximum lengths of code words. Let $n_j = |C_j|$ (the number of elements of C_j). If f is prefix, then we have

$$\begin{aligned} &n_1 + \cdots + n_l = n \text{ and} \\ &n_l \leq r^l - n_1 r^{l-1} - \cdots - n_{l-1} r. \end{aligned} \qquad [*]$$

The first relation is clear, as $C_1 \cup \cdots \cup C_l$ is the set of code words and there are n of these. As for the inequality, note that there are precisely $n_j r^{l-j}$ elements in $\mathcal{C}^l$ having an element of C_j as prefix (we can add any $l - j$ symbols to any of the n_j elements of C_j). Excluding all these elements, we are left with a subset of $\mathcal{C}^l$ with $r^l - n_1 r^{l-1} - \cdots - n_{l-1} r$ elements that must contain C_l, and so $[*]$ is also clear.

Now it is interesting to see that the converse is also true.

Proposition. If $n_1, n_2, \dots, n_l$ are non-negative integers that satisfy the conditions [$*$], then there is a prefix encoding $f: \mathcal{A} \to \mathcal{C}^*$ that has exactly n_j code words of length j for $j = 1, \dots, l$.

Proof. From [$*$], and the fact that the n_j are non-negative, we get that

$$n_j \le r^j - n_1 r^{j-1} - \cdots - n_{j-1} r \ \ (j = l, \dots, 1) \qquad [*_j]$$

Notice that from $0 \le r^j - n_1 r^{j-1} - \cdots - n_{j-1} r$ we get [$*_{j-1}$] (divide by r and move n_{j-1} to the left hand side).

We are going to construct subsets $C_j \subseteq \mathcal{C}^j$ ($j = 1, \dots, l$), with

(a) $|C_j| = n_j$, and

(b) for $j' < j$, no element of $C_{j'}$ is a prefix of an element of C_j ($j = 2, \dots, l$).

As $C_1 \subseteq \mathcal{C}$ we take any subset with $|C_1| = n_1$ (this is certainly possible because [$*_1$] tells us that $n_1 \le r$).

Suppose then that for some $j > 1$, $j \leq l$, we have constructed $C_1, \dots, C_{j-1}$ with the properties (*a*) and (*b*). To add C_j to the list, first note that for any $1 \leq j' < j$ there are precisely $n_{j'} r^{j-j'}$ elements in $\mathcal{C}^j$ that have an element of $C_{j'}$ as a prefix (we can add any $j - j'$ symbols to any of the $n_{j'}$ elements of $C_{j'}$). We are left with a subset $S_j \subseteq \mathcal{C}^j$ with at least

$$r^j - n_1 r^{j-1} - \cdots - n_{j-1} r$$

elements such that for $j' < j$ no element of $C_{j'}$ is a prefix of any of the elements of S_j. Then it is enough to take as C_j any subset of n_j elements of S_j, which is possible because $|S_j| \geq n_j$ (by $[*_j]$).

Having the sets C_j with the properties (*a*) and (*b*), it is easy to construct an encoding $f\colon \mathcal{A} \to \mathcal{C}^*$ with $C_1 \cup \cdots \cup C_l$ as set of code words, and this encoding is instantaneous.

E.3.1. Show that the inequality $[*_j]$ is strict except maybe for $j = l$.

Remark. The preceding considerations are often stated as follows:
There exists a prefix encoding $f\colon \mathcal{A} \to \mathcal{C}^*$ with word lengths $l_1, \dots, l_n$ iff

$$\frac{1}{r^{l_1}} + \cdots + \frac{1}{r^{l_n}} \leq 1 \text{ (Kraft's inequality).}$$

Note that this inequality is equivalent, setting $l = \max(l_j)$, to

$$r^l \geq r^{l-l_1} + \cdots + r^{l-l_n}.$$

Since on the right hand side there are n_j terms of the form r^{l-j}, the last inequality can be written as

$$r^l \geq n_1 r^{l-1} + \cdots + n_{l-1} r + n_l,$$

which is equivalent to $[*]$.

Remark (Kraft-MacMillan inequality). If we have an encoding $f\colon \mathcal{A} \to \mathcal{C}^*$ with word lengths $l_1, \dots, l_n$, then Kraft's inequality is still satisfied (McMillan theorem[N3]) and hence there is a prefix encoding with the same word lengths (Moral: *with prefix encodings we can do as much as with ud's*).

3.3. The noiseless coding theorem

Given and encoding $f\colon \mathcal{A} \to \mathcal{C}^*$, let $l_i = |f(a_i)|$ (the length of $f(a_i)$), the *mean length*, denoted $\ell = \ell_f$, is defined as

$$\ell = p_1 l_1 + \cdots + p_n l_n.$$

Theorem (Shannon). Let $f\colon \mathcal{A} \to \mathcal{C}^*$ be an encoding (so ud). Then $\ell \geq H/\log_2(r)$, where $H = -\sum_j p_j \log_2(p_j)$ is the entropy of the source. Moreover, there exists a encoding with $\ell < 1 + H/\log_2(r)$.

Proof. Set $S = \frac{1}{r^{l_1}} + \cdots + \frac{1}{r^{l_n}}$ and $q_j = \frac{1}{Sr^{l_j}}$. Then $S \leq 1$, by the Kraft-Macmillan inequality, and $q_1 + \cdots + q_n = 1$. By Gibbs lemma,

$$H = -\sum_j p_j \log_2\left(p_j\right) \leq -\sum_j p_j \log_2(q_j).$$

Since $\log_2(q_j) = -l_j \log_2(r) - \log_2(S)$, and $\log_2(S) \leq 0$, we obtain

$$H \leq \ell \log_2(r) + \log_2(S) \leq \ell \log_2(r).$$

Note: The equality $H = \ell \log_2(r)$ happens iff $S = 1$ and $p_j = 1/r^{l_j}$.

To prove the second part, let l_j be the least integer such that $r^{l_j} \geq 1/p_j$. Then $p_j \geq r^{-l_j}$ and $\sum_j r^{-l_j} \leq \sum_j p_j = 1$. Thus the l_j satisfy the Kraft-McMillan inequality and hence there is a prefix (*sic*) encoding $f\colon \mathcal{A} \to \mathcal{C}^*$ with code lengths $l_1, \dots, l_n$. But $l_j < 1 - \log_2(p_j) / \log_2(r)$, by E.2.2, and hence $\ell = \sum_j p_j l_j < 1 + H/\log_2(r)$.

E.3.2. Show that the least integer such that $r^{l_j} \geq 1/p_j$ is

$$\lceil -\log_2(p_j) / \log_2(r) \rceil.$$

In particular, $l_j < 1 - \log_2(p_j) / \log_2(r)$.

E.3.3. Use the Kraft-McMillan inequality to show that the set

$$C = \{0, 01, 11, 101\}$$

is not a code and find a binary string that can be decomposed in two different ways as a concatenation of elements of C.

3.4. Optimal encodings and the Huffman's algorithm

An encoding $f\colon \mathcal{A} \to \mathcal{C}^*$ is said to be *optimal*, or *compact*, if ℓ has the minimum possible value. In the search for such encodings, we know that it is enough to look for prefix encodings. Moreover, ℓ must satisfy the constraints given by Shannon's source coding theorem:

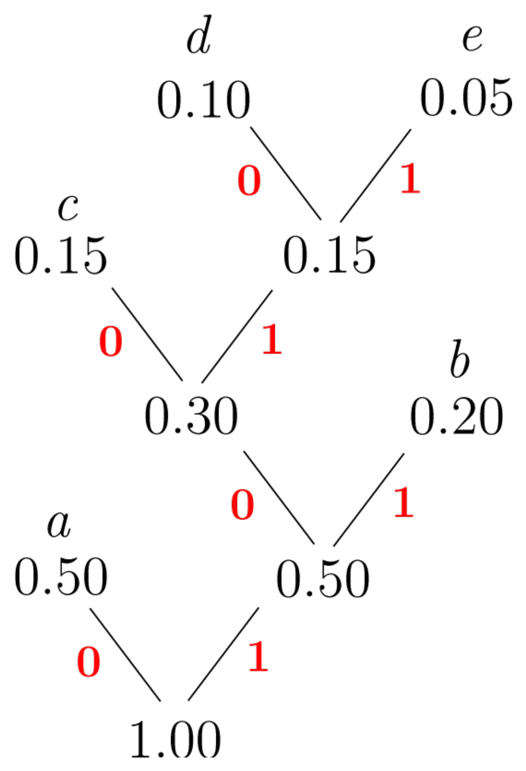
$$H/\log_2(r) \leq \ell < 1 + H/\log_2(r).$$

Optimal prefix binary encodings $f\colon \mathcal{A} \to \{0,1\}^*$ (*Huffman encodings*) were constructed by Huffman in 1952. The purpose of this section is to describe in detail his procedure.

We will explain first how the algorithm works in two examples. Given the source symbols $(a_1, \dots, a_n)$ and their probabilities $(p_1, \dots, p_n)$, we will build a binary tree of depth $n-1$ and having $a_1, \dots, a_n$ as leaves. The left (right) edges are labeled 0 (1). The encoding of a symbol is done by reading the labels found in going from the root to the symbol leaf.

Example I (cf. Welsh-1988, p. 22-23). We take the alphabet $\{a, b, c, d, e\}$, with probabilities $\{0.50, 0.20, 0.15, 0.10, 0.05\}$. In this case, Huffman's algorithm constructs the tree shown in the figure and hence it outputs the encoding

$$a \to 0,\ b \to 11,\ c \to 100,$$
$$d \to 1010,\ e \to 1011.$$

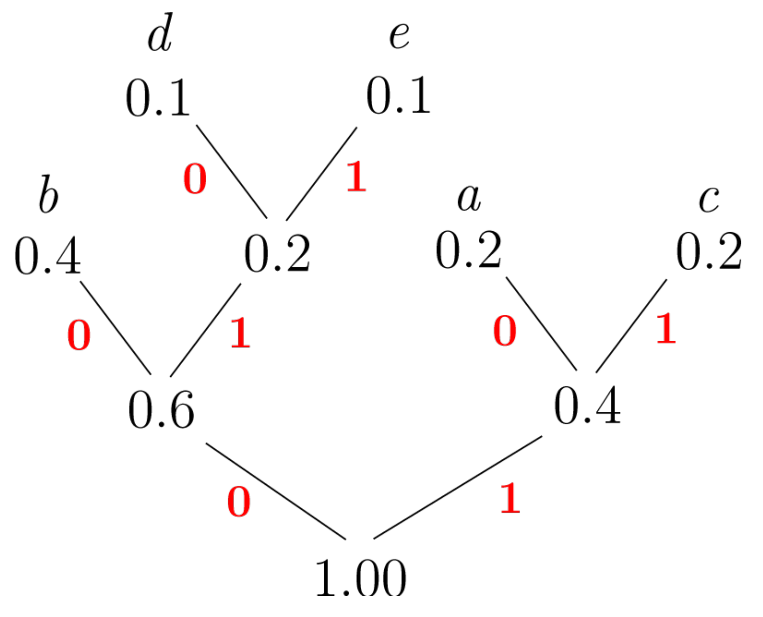


Huffman: Example I

We start with the two leaves with smallest probability (d and e), with the highest to the left. We add a node, labeled $0.15 = 0.10 + 0.05$, having d and e as children. Now the smallest probabilities are $p(c) = 0.15$ and the node 0.15 just constructed, and to them we attach a new node, 0.30. We keep c to the left because it was known earlier. Then we proceed with $0.30 > 0.20 = p(b)$, hence b is placed to the right, and finally we end with the node 1.00 (the root).

Example II (cf. Blelloch-2010, p. 15-16). We take again the alphabet $\{a, b, c, d, e\}$, with probabilities $\{0.2, 0.4, 0.2, 0.1, 0.1\}$. In this case, Huffman's algorithm constructs the tree shown in the figure and hence it outputs the encoding

$$a \to 10, b \to 00, c \to 11,$$
$$d \to 010, e \to 011.$$



Huffman: Example II

The smallest probabilities are $p(d) = p(e) = 0.1$. We proceed with them as in the previous example, yielding the node 0.2 that ties with a and c. Since these were known before, we create the node 0.4, that ties with b. Now we combine $0.4 = p(b)$, known before, with the 0.2, which is now the smallest value, yielding the node 0.6.

Remark. If we do not obey the precedence rules explained in the examples, we still get an optimal prefix code. With the precedence rules, it turns out that the lengths of the code words have the lowest possible variance.**N4**

Algorithm (Huffman). *Input*: the symbol probabilities. *Output*: a binary tree with nodes labeled with numbers in the range $(0,1]$ and edges labeled with 0 or 1.

Description. *Start*. Consider each symbol as a depth 0 tree labeled with its probability.

Repeat. Do until a single tree remains (which will happen in $< n$ steps):

- select two trees with the lowest weight roots (say w and w').
- combine the two trees into a single tree by adding a new root with weight $w + w'$, and making the two trees its children.

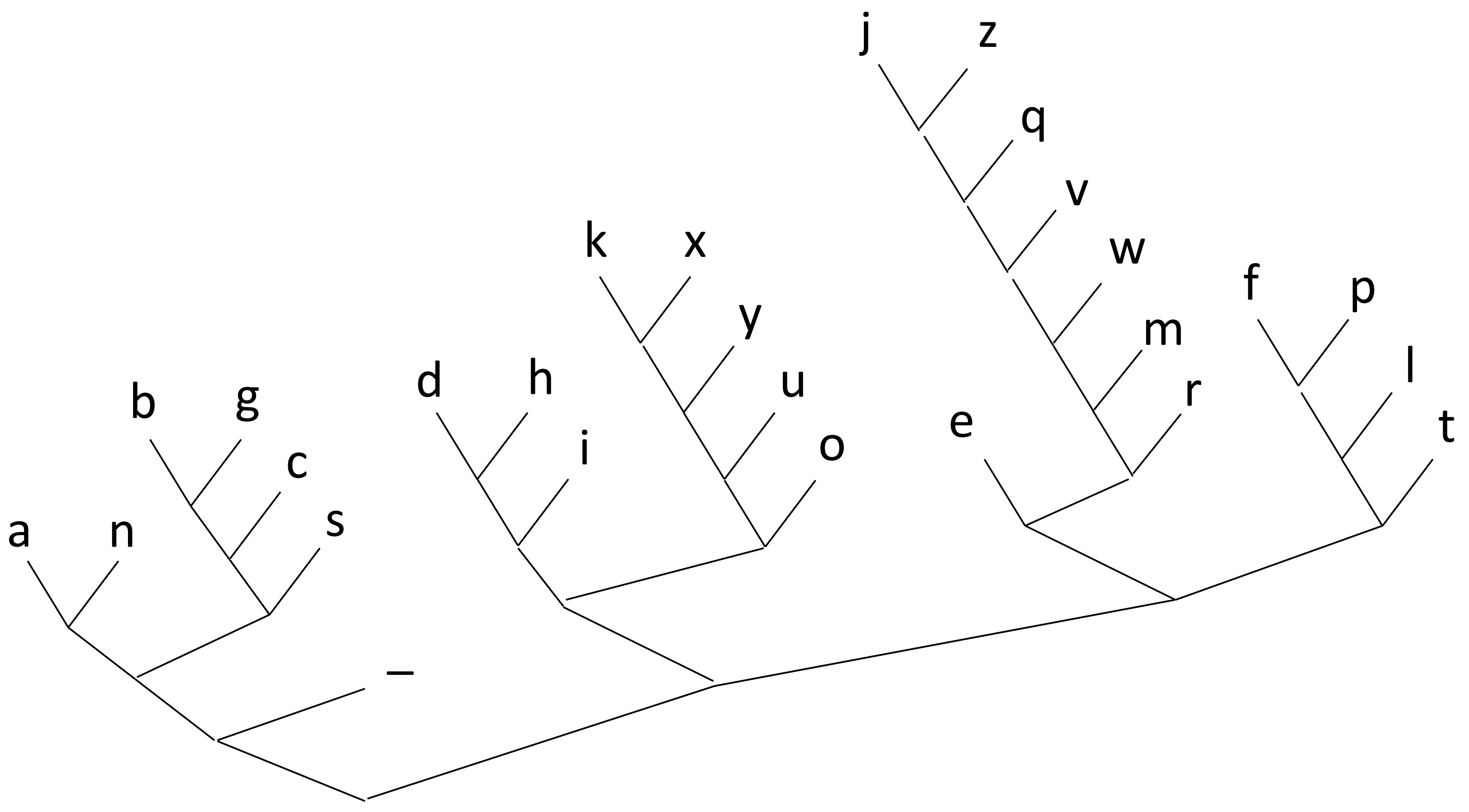
For a proof of the correctness of the algorithm, including the strengthening explained in the note, see Blelloch-2010.

Example III (cf. [MacKay-2003])

Data for monogram English

$L_j = \log_2(1/p_j)$

a_j	p_j	L_j	l_j	$f(a_j)$	a_j	p_j	L_j	l_j	$f(a_j)$
a	0.0575	4.1	4	0000	o	0.0689	3.9	4	1011
b	0.0128	6.3	6	001000	p	0.0192	5.7	6	111001
c	0.0263	5.2	5	00101	q	0.0008	10.3	9	110100001
d	0.0285	5.1	5	10000	r	0.0508	4.3	5	11011
e	0.0913	3.5	4	1100	s	0.0567	4.1	4	0011
f	0.0173	5.9	6	111000	t	0.0706	3.8	4	1111
g	0.0133	6.2	6	001001	u	0.0334	4.9	5	10101
h	0.0313	5.0	5	10001	v	0.0069	7.2	8	11010001
i	0.0599	4.1	4	1001	w	0.0119	6.4	7	1101001
j	0.0006	10.7	10	1101000000	x	0.0073	7.1	7	1010001
k	0.0084	6.9	7	1010000	y	0.0164	5.9	6	101001
l	0.0335	4.9	5	11101	z	0.0007	10.4	10	1101000001
m	0.0235	5.4	6	110101	_	0.1928	2.4	2	01
n	0.0596	4.1	4	0001					



Huffman code for monogram English

Remark. The entropy of monogram English is $H = 4.1094$, while the average codeword length is $4.1462 \geq H$.

Notes

N1 (p. 3). If $f(a_{i_1})f(a_{i_2})\cdots f(a_{i_k}) = f(a_{i'_1})f(a_{i'_2})\cdots f(a_{i'_k})$, then either $f(a_{i_1})$ is a prefix of $f(a_{i'_1})$ or $f(a_{i'_1})$ is a prefix of $f(a_{i_1})$ or $f(a_{i_1}) = f(a_{i'_1})$. If the code is instantaneous, only the last possibility can occur, and so $a_{i_1} = a_{i'_1}$. Now the claim follows easily by induction.

N2 (p. 3). Work backwards from the end of a given binary string. If it ends with 0, this 0 is uniquely decoded as x and we can proceed by induction. If it ends with 1 and the previous bit is 0, the last two bits are uniquely decoded as y, and we can again proceed by induction. If the last two bits are 11, then the binary string is not decodable.

N3 (p. 8). Let $f\colon \mathcal{A} \to \mathcal{C}^*$ be an encoding with word lengths $l_1, \ldots, l_n$. Then we can write, for any positive integer m,

$$\left(\frac{1}{r^{l_1}} + \cdots + \frac{1}{r^{l_n}}\right)^m = \frac{v_1}{r} + \frac{v_2}{r^2} + \cdots + \frac{v_{lm}}{r^{lm}} \quad ,$$

where $v_j \geq 0$ is the number of ways in which j can be decomposed into a sum of m integers from the set $\{l_1, \dots, l_n\}$. If the code is ud, then we clearly must have $v_j \leq r^j$, and hence

$$\left(\frac{1}{r^{l_1}} + \cdots + \frac{1}{r^{l_n}}\right)^m \leq lm, \text{ or}$$

$$\frac{1}{r^{l_1}} + \cdots + \frac{1}{r^{l_n}} \leq l^{1/m} s^{1/m},$$

and letting m go to ∞ we get Kraft's inequality.

N4 (p. 14). The variance of the code-lengths is $\sigma = \sum_j p_j \left(l_j - \ell\right)^2$. Remember that $\ell = \sum_j p_j l_j$. In the example, σ is equal to 1.7895.

References

[Blelloch-2010] Guy E. Blelloch: *Introduction to data compression*. http://www.cs.cmu.edu/~guyb/realworld/compression.pdf.

[MacKay-2003] David J.C. MacKay: *Information theory, inference and learning algo-rithms.* Cambridge university press, 2003. xii + 628p.