1
Essentials of Information Theory

Throughout the book, the symbol $\mathbb{P}$ denotes various probability distributions. In particular, in Chapter 1, $\mathbb{P}$ refers to the probabilities for sequences of random variables characterising sources of information. As a rule, these are sequences of independent and identically distributed random variables or discrete-time Markov chains; namely, $\mathbb{P}(U_1 = u_1, \ldots, U_n = u_n)$ is the joint probability that random variables $U_1, \ldots, U_n$ take values $u_1, \ldots, u_n$, and $\mathbb{P}(V = v | U = u, W = w)$ is the conditional probability that a random variable V takes value v, given that random variables U and W take values u and w, respectively. Likewise, $\mathbb{E}$ denotes the expectation with respect to $\mathbb{P}$.

The symbols p and P are used to denote various probabilities (and probability-related objects) loosely. The symbol $\sharp A$ denotes the cardinality of a finite set A. The symbol $\mathbf{1}$ stands for an indicator function. We adopt the following notation and formal rules for logarithms: $\ln = \log_e$, $\log = \log_2$, and for all $b > 1$: $0 \cdot \log_b 0 = 0 \cdot \log_b \infty = 0$. Next, given $x > 0$, $\lfloor x \rfloor$ and $\lceil x \rceil$ denote the maximal integer that is no larger than x and the minimal integer that is no less than x, respectively. Thus, $\lfloor x \rfloor \leq x \leq \lceil x \rceil$; equalities hold here when x is a positive integer ($\lfloor x \rfloor$ is called the integer part of x.)

The abbreviations LHS and RHS stand, respectively, for the left-hand side and the right-hand side of an equation.

1.1 Basic concepts. The Kraft inequality. Huffman's encoding

A typical scheme used in information transmission is as follows:

$$\boxed{\text{A message source}} \to \boxed{\text{an encoder}} \to \boxed{\text{a channel}}$$
$$\to \boxed{\text{a decoder}} \to \boxed{\text{a destination}}$$

Example 1.1.1 (a) A message source: a Cambridge college choir.
(b) An encoder: a BBC recording unit. It translates the sound to a binary array and writes it to a CD track. The CD is then produced and put on the market.
(c) A channel: a customer buying a CD in England and mailing it to Australia. The channel is subject to 'noise': possible damage (mechanical, electrical, chemical, etc.) incurred during transmission (transportation).
(d) A decoder: a CD player in Australia.
(e) A destination: an audience in Australia.
(f) The goal: to ensure a high-quality sound despite damage.

In fact, a CD can sustain damage done by a needle while making a neat hole in it, or by a tiny drop of acid (you are not encouraged to make such an experiment!). In technical terms, typical goals of information transmission are:

(i) fast encoding of information,
(ii) easy transmission of encoded messages,
(iii) effective use of the channel available (i.e. maximum transfer of information per unit time),
(iv) fast decoding,
(v) correcting errors (as many as possible) introduced by noise in the channel.

As usual, these goals contradict each other, and one has to find an optimal solution. This is what the chapter is about. However, do not expect perfect solutions: the theory that follows aims mainly at providing knowledge of the basic principles. A final decision is always up to the individual (or group) responsible.

A large part of this section (and the whole of Chapter 1) will deal with *encoding* problems. The aims of encoding are:

(1) compressing data to reduce redundant information contained in a message,
(2) protecting the text from unauthorised users,
(3) enabling errors to be corrected.

We start by studying *sources* and *encoders*. A source emits a sequence of letters (or symbols),

$$u_1\, u_2\, \ldots\, u_n \ldots, \tag{1.1.1}$$

where $u_j \in I$, and $I(= I_m)$ is an m-element set often identified as $\{1, \ldots, m\}$ (a source alphabet). In the case of literary English, $m = 26 + 7$, 26 letters plus 7 punctuation symbols: . , : ; – ( ). (Sometimes one adds ? ! ' ' and "). Telegraph English corresponds to $m = 27$.

A common approach is to consider (1.1.1) as a *sample* from a random source, i.e. a sequence of random variables

$$U_1, U_2, \ldots, U_n, \ldots \tag{1.1.2}$$

and try to develop a theory for a reasonable class of such sequences.

Example 1.1.2 (a) The simplest example of a random source is a sequence of independent and identically distributed random variables (IID random variables):

$$\mathbb{P}(U_1 = u_1,\ U_2 = u_2, \ldots, U_k = u_k) = \prod_{j=1}^{k} p(u_j), \tag{1.1.3a}$$

where $p(u) = \mathbb{P}(U_j = u)$, $u \in I$, is the marginal distribution of a single variable. A random source with IID symbols is often called a *Bernoulli source*.

A particular case where $p(u)$ does not depend on $u \in U$ (and hence equals $1/m$) corresponds to the equiprobable Bernoulli source.
(b) A more general example is a Markov source where the symbols form a discrete-time Markov chain (DTMC):

$$\mathbb{P}(U_1 = u_1,\ U_2 = u_2, \ldots,\ U_k = u_k) = \lambda(u_1) \prod_{j=1}^{k-1} P(u_j, u_{j+1}), \tag{1.1.3b}$$

where $\lambda(u) = \mathbb{P}(U_1 = u)$, $u \in I$, are the initial probabilities and $P(u,u') = \mathbb{P}(U_{j+1} = u' | U_j = u)$, $u, u' \in I$, are transition probabilities. A Markov source is called *stationary* if $\mathbb{P}(U_j = u) = \lambda(u)$, $j \geq 1$, i.e. $\lambda = \{\lambda(u), u = 1, \ldots, m\}$ is an invariant row-vector for matrix $P = \{P(u,v)\}$: $\sum\limits_{u \in I} \lambda(u) P(u,v) = \lambda(v)$, $v \in I$, or, shortly, $\lambda P = \lambda$.
(c) A 'degenerated' example of a Markov source is where a source emits repeated symbols. Here,

$$\begin{gathered} \mathbb{P}(U_1 = U_2 = \cdots = U_k = u) = p(u),\ \ u \in I, \\ \mathbb{P}(U_k \neq U_{k'}) = 0,\ \ 1 \leq k < k', \end{gathered} \tag{1.1.3c}$$

where $0 \leq p(u) \leq 1$ and $\sum\limits_{u \in I} p(u) = 1$.

An initial piece of sequence (1.1.1)

$$\mathbf{u}^{(n)} = (u_1, u_2, \ldots, u_n) \quad \text{or, more briefly,} \quad \mathbf{u}^{(n)} = u_1 u_2 \ldots u_n$$

is called a (source) sample *n-string*, or *n-word* (in short, a string or a word), with digits from I, and is treated as a 'message'. Correspondingly, one considers a random n-string (a random message)

$$\mathbf{U}^{(n)} = (U_1, U_2, \ldots, U_n) \quad \text{or, briefly,} \quad \mathbf{U}^{(n)} = U_1 U_2 \ldots U_n.$$

An encoder (or coder) uses an alphabet $J (= J_q)$ which we typically write as $\{0, 1, \ldots, q-1\}$; usually the number of encoding symbols $q < m$ (or even $q \ll m$); in many cases $q = 2$ with $J = \{0, 1\}$ (a binary coder). A *code* (also *coding*, or *encoding*) is a map, f, that takes a symbol $u \in I$ into a finite string, $f(u) = x_1 \dots x_s$, with digits from J. In other words, f maps I into the set J^* of all possible strings:

$$f : I \to J^* = \bigcup_{s \geq 1} \big(J \times \cdots \,(s \text{ times})\, \times J \big).$$

Strings $f(u)$ that are images, under f, of symbols $u \in I$ are called *codewords* (in code f). A code has (constant) length N if the value s (the length of a codeword) equals N for all codewords. A message $\mathbf{u}^{(n)} = u_1 u_2 \dots u_n$ is represented as a concatenation of codewords

$$f(\mathbf{u}^{(n)}) = f(u_1) f(u_2) \dots f(u_n);$$

it is again a string from J^*.

Definition 1.1.3 We say that a code is *lossless* if $u \neq u'$ implies that $f(u) \neq f(u')$. (That is, the map $f : I \to J^*$ is one-to-one.) A code is called *decipherable* if any string from J^* is the image of at most one message. A string x is a *prefix* in another string y if $y = xz$, i.e. y may be represented as a result of a concatenation of x and z. A code is *prefix-free* if no codeword is a prefix in any other codeword (e.g. a code of constant length is prefix-free).

A prefix-free code is decipherable, but not vice versa:

Example 1.1.4 A code with three source letters $1, 2, 3$ and the binary encoder alphabet $J = \{0, 1\}$ given by

$$f(1) = 0, \;\; f(2) = 01, \;\; f(3) = 011$$

is decipherable, but not prefix-free.

Theorem 1.1.5 (The Kraft inequality) *Given positive integers $s_1, \dots, s_m$, there exists a decipherable code $f : I \to J^*$, with codewords of lengths $s_1, \dots, s_m$, iff*

$$\sum_{i=1}^{m} q^{-s_i} \leq 1. \tag{1.1.4}$$

Furthermore, under condition (1.1.4) *there exists a prefix-free code with codewords of lengths $s_1, \dots, s_m$.*

Proof (I) Sufficiency. Let (1.1.4) hold. Our goal is to construct a prefix-free code with codewords of lengths $s_1, \dots, s_m$. Rewrite (1.1.4) as

$$\sum_{l=1}^{s} n_l q^{-l} \leq 1, \tag{1.1.5}$$

or

$$n_s q^{-s} \le 1 - \sum_{l=1}^{s-1} n_l q^{-l},$$

where n_l is the number of codewords of length l and $s = \max s_i$. Equivalently,

$$n_s \le q^s - n_1 q^{s-1} - \cdots - n_{s-1} q. \tag{1.1.6a}$$

Since $n_s \ge 0$, deduce that

$$n_{s-1} q \le q^s - n_1 q^{s-1} - \cdots - n_{s-2} q^2,$$

or

$$n_{s-1} \le q^{s-1} - n_1 q^{s-2} - \cdots - n_{s-2} q. \tag{1.1.6b}$$

Repeating this argument yields subsequently

$$\begin{array}{ccccccc} n_{s-2} & \le & q^{s-2} - n_1 q^{s-3} & - & \dots & -n_{s-3} q \\ \vdots & & \vdots & & & \vdots \\ n_2 & \le & q^2 - n_1 q & & & \end{array} \tag{1.1.6.s−1}$$

$$n_1 \le q. \tag{1.1.6.s}$$

Observe that actually either $n_{i+1} = 0$ or n_i is less than the RHS of the inequality, for all $i = 1, \dots, s-1$ (by definition, $n_s \ge 1$ so that for $i = s-1$ the second possibility occurs). We can perform the following construction. First choose n_1 words of length 1, using distinct symbols from J: this is possible in view of (1.1.6.s). It leaves $(q - n_1)$ symbols unused; we can form $(q - n_1)q$ words of length 2 by appending a symbol to each. Choose n_2 codewords from these: we can do so in view of (1.1.6.s−1). We still have $q^2 - n_1 q - n_2$ words unused: form n_3 codewords, etc. In the course of the construction, no new word contains a previous codeword as a prefix. Hence, the code constructed is prefix-free.

(II) Necessity. Suppose there exists a decipherable code in J^* with codeword lengths $s_1, \dots, s_m$. Set $s = \max s_i$ and observe that for any positive integer r

$$\left(q^{-s_1} + \cdots + q^{-s_m}\right)^r = \sum_{l=1}^{rs} b_l q^{-l}$$

where b_l is the number of ways r codewords can be put together to form a string of length l.

Because of decipherability, these strings must be distinct. Hence, we must have $b_l \leq q^l$, as q^l is the total number of l-strings. Then

$$\left(q^{-s_1}+\cdots+q^{-s_m}\right)^r \leq rs,$$

and

$$q^{-s_1}+\cdots+q^{-s_m} \leq r^{1/r}s^{1/r} = \exp\left[\frac{1}{r}(\log r+\log s)\right].$$

This is true for any r, so take $r \to \infty$. The RHS goes to 1. □

Remark 1.1.6 A given code obeying (1.1.4) is not necessarily decipherable.

Leon G. Kraft introduced inequality (1.1.4) in his MIT PhD thesis in 1949.

One of the principal aims of the theory is to find the ‘best’ (that is, the shortest) decipherable (or prefix-free) code. We now adopt a probabilistic point of view and assume that symbol $u \in I$ is emitted by a source with probability $p(u)$:

$$\mathbb{P}(U_k = u) = p(u).$$

[At this point, there is no need to specify a joint probability of more than one subsequently emitted symbol.]

Recall, given a code $f : I \mapsto J^*$, we encode a letter $i \in I$ by a prescribed codeword $f(i) = x_1 \ldots x_{s(i)}$ of length $s(i)$. For a random symbol, the generated codeword becomes a random string from J^*. When f is lossless, the probability of generating a given string as a codeword for a symbol is precisely $p(i)$ if the string coincides with $f(i)$ and 0 if there is no letter $i \in I$ with this property. If f is not one-to-one, the probability of a string equals the sum of terms $p(i)$ for which the codeword $f(i)$ equals this string. Then the length of a codeword becomes a *random variable*, S, with the probability distribution

$$\mathbb{P}(S = s) = \sum_{1\leq i\leq m} \mathbf{1}(s(i) = s)p(i). \tag{1.1.7}$$

We are looking for a decipherable code that minimises the expected word-length:

$$\mathbb{E}S = \sum_{s\geq 1} s\mathbb{P}(S = s) = \sum_{i=1}^{m} s(i)p(i).$$

The following problem therefore arises:

$$\begin{aligned} &\text{minimise } g(s(1),\ldots,s(m)) = \mathbb{E}S \\ &\text{subject to } \sum_i q^{-s(i)} \leq 1 \text{ (Kraft)} \\ &\text{with } s(i) \text{ positive integers.} \end{aligned} \tag{1.1.8}$$

Theorem 1.1.7 *The optimal value for problem* (1.1.8) *is lower-bounded as follows:*

$$\min \mathbb{E}S \geq h_q(p(1),\dots,p(m)), \tag{1.1.9}$$

where

$$h_q(p(1),\dots,p(m)) = -\sum_i p(i)\log_q p(i). \tag{1.1.10}$$

Proof The algorithm (1.1.8) is an integer-valued optimisation problem. If we drop the condition that $s(1),\dots,s(m) \in \{1,2,\dots\}$, replacing it with a 'relaxed' constraint $s(i) > 0$, $1 \leq i \leq m$, the Lagrange sufficiency theorem could be used. The Lagrangian reads

$$\mathscr{L}(s(1),\dots,s(m),z;\lambda) = \sum_i s(i)p(i) + \lambda(1 - \sum_i q^{-s(i)} - z)$$

(here, $z \geq 0$ is a slack variable). Minimising $\mathscr{L}$ in $s_1,\dots,s_m$ and z yields

$$\lambda < 0, \quad z = 0, \text{ and } \frac{\partial \mathscr{L}}{\partial s(i)} = p(i) + q^{-s(i)}\lambda \ln q = 0,$$

whence

$$-\frac{p(i)}{\lambda \ln q} = q^{-s(i)}, \quad \text{i.e. } s(i) = -\log_q p(i) + \log_q(-\lambda \ln q),\ 1 \leq i \leq m.$$

Adjusting the constraint $\sum\limits_i q^{-s(i)} = 1$ (the slack variable $z = 0$) gives

$$\sum_i p(i)/(-\lambda \ln q) = 1, \text{ i.e. } -\lambda \ln q = 1.$$

Hence,

$$s(i) = -\log_q p(i), \quad 1 \leq i \leq m,$$

is the (unique) optimiser for the relaxed problem, giving the value h_q from (1.1.10). The relaxed problem is solved on a larger set of variables $s(i)$; hence, its minimal value does not exceed that in the original one. □

Remark 1.1.8 The quantity h_q defined in (1.1.10) plays a central role in the whole of information theory. It is called the q-ary entropy of the probability distribution $(p(x), x \in I)$ and will emerge in a great number of situations. Here we note that the dependence on q is captured in the formula

$$h_q(p(1),\dots,p(m)) = \frac{1}{\log q} h_2(p(1),\dots,p(m))$$

where h_2 stands for the binary entropy:

$$h_2(p(1),\dots,p(m)) = -\sum_i p(i)\log p(i). \tag{1.1.11}$$

Worked Example 1.1.9 (a) *Give an example of a lossless code with alphabet J_q which does not satisfy the Kraft inequality. Give an example of a lossless code with the expected code-length strictly less than $h_q(X)$.*

(b) *Show that the 'Kraft sum' $\sum\limits_i q^{-s(i)}$ associated with a lossless code may be arbitrarily large (for sufficiently large source alphabet).*

Solution (a) Consider the alphabet $I = \{0,1,2\}$ and a lossless code f with $f(0) = 0, f(1) = 1, f(2) = 00$ and codeword-lengths $s(0) = s(1) = 1, s(2) = 2$. Obviously, $\sum\limits_{x \in I} 2^{-s(x)} = 5/4$, violating the Kraft inequality. For a random variable X with $p(0) = p(1) = p(2) = 1/3$ the expected codeword-length $\mathbb{E}s(X) = 4/3 < h(X) = \log 3 = 1.585$.

(b) Assume that the alphabet size $m = \sharp I = 2(2^L - 1)$ for some positive integer L. Consider the lossless code assigning to the letters $x \in I$ the codewords $0, 1, 00, 01, 10, 11, 000, \ldots$, with the maximum codeword-length L. The Kraft sum is

$$\sum_{x \in I} 2^{-s(x)} = \sum_{l \le L} \sum_{x:s(x)=l} 2^{-s(x)} = \sum_{l \le L} 2^l \times 2^{-l} = L,$$

which can be made arbitrarily large. □

The assertion of Theorem 1.1.7 is further elaborated in

Theorem 1.1.10 (Shannon's noiseless coding theorem (NLCT)) *For a random source emitting symbols with probabilities $p(i) > 0$, the minimal expected codeword-length for a decipherable encoding in alphabet J_q obeys*

$$h_q \le \min \mathbb{E}S < h_q + 1, \tag{1.1.12}$$

where $h_q = -\sum\limits_i p(i) \log_q p(i)$ is the q-ary entropy of the source; see (1.1.10).

Proof The LHS inequality is established in (1.1.9). For the RHS inequality, let $s(i)$ be a positive integer such that

$$q^{-s(i)} \le p(i) < q^{-s(i)+1}.$$

The non-strict bound here implies $\sum\limits_i q^{-s(i)} \le \sum\limits_i p(i) = 1$, i.e. the Kraft inequality. Hence, there exists a decipherable code with codeword-lengths $s(1), \ldots, s(m)$. The strict bound implies

$$s(i) < -\frac{\log p(i)}{\log q} + 1,$$

and thus

$$\mathbb{E}S < -\frac{\sum_i p(i)\log p(i)}{\log q} + \sum_i p(i) = \frac{h}{\log q} + 1.$$

$\square$

Example 1.1.11 An instructive application of Shannon's NLCT is as follows. Let the size m of the source alphabet equal 2^k and assume that the letters $i = 1, \dots, m$ are emitted equiprobably: $p(i) = 2^{-k}$. Suppose we use the code alphabet $J_2 = \{0, 1\}$ (binary encoding). With the binary entropy $h_2 = -\log 2^{-k} \sum_{1 \le i \le 2^k} 2^{-k} = k$, we need, on average, at least k binary digits for decipherable encoding. Using a term *bit* for a unit of entropy, we say that on average the encoding requires at least k bits.

Moreover, the NLCT leads to a *Shannon–Fano encoding* procedure: we fix positive integer codeword-lengths $s(1), \dots, s(m)$ such that $q^{-s(i)} \le p(i) < q^{-s(i)+1}$, or, equivalently,

$$-\log_q p(i) \le s(i) < -\log_q p(i) + 1;\ \text{that is, } s(i) = \left\lceil -\log_q p(i) \right\rceil. \qquad (1.1.13)$$

Then construct a prefix-free code, from the shortest $s(i)$ upwards, ensuring that the previous codewords are not prefixes. The Kraft inequality guarantees enough room. The obtained code may not be optimal but has the mean codeword-length satisfying the same inequalities (1.1.13) as an optimal code.

Optimality is achieved by *Huffman's encoding* $f_m^{\mathrm{H}}\colon I_m \mapsto J_q^*$. We first discuss it for binary encodings, when $q = 2$ (i.e. $J = \{0, 1\}$). The algorithm constructs a binary tree, as follows.

(i) First, order the letters $i \in I$ so that $p(1) \ge p(2) \ge \dots \ge p(m)$.
(ii) Assign symbol 0 to letter $m - 1$ and 1 to letter m.
(iii) Construct a reduced alphabet $I_{m-1} = \{1, \dots, m-2, (m-1, m)\}$, with probabilities

$$p(1), \dots,\ p(m-2),\ p(m-1) + p(m).$$

Repeat steps (i) and (ii) with the reduced alphabet, etc. We obtain a binary tree. For an example of Huffman's encoding for $m = 7$ see Figure 1.1.

The number of branches we must pass through in order to reach a root i of the tree equals $s(i)$. The tree structure, together with the identification of the roots as source letters, guarantees that encoding is prefix-free. The optimality of binary Huffman encoding follows from the following two simple lemmas.

$m = 7$

i	p_i	$f(i)$	s_i
1	.5	0	1
2	.15	100	3
3	.15	101	3
4	.1	110	3
5	.05	1110	4
6	.025	11110	5
7	.025	11111	5

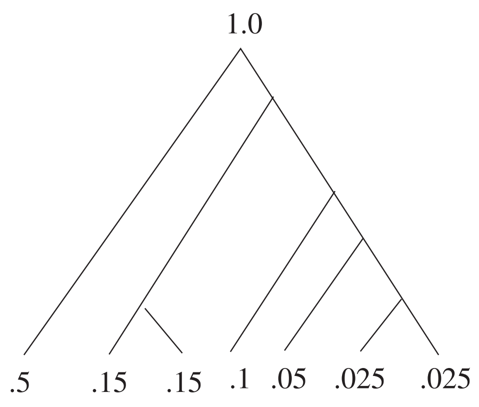

Figure 1.1

Lemma 1.1.12 *Any optimal prefix-free binary code has the codeword-lengths reverse-ordered versus probabilities:*

$$p(i) \geq p(i') \quad \textit{implies} \quad s(i) \leq s(i'). \tag{1.1.14}$$

Proof If not, we can form a new code, by swapping the codewords for i and i'. This shortens the expected codeword-length and preserves the prefix-free property. □

Lemma 1.1.13 *In any optimal prefix-free binary code there exist, among the codewords of maximum length, precisely two agreeing in all but the last digit.*

Proof If not, then either (i) there exists a single codeword of maximum length, or (ii) there exist two or more codewords of maximum length, and they all differ before the last digit. In both cases we can drop the last digit from some word of maximum length, without affecting the prefix-free property. □

Theorem 1.1.14 *Huffman's encoding is optimal among the prefix-free binary codes.*

Proof The proof proceeds with induction in m. For $m = 2$, the Huffman code f_2^{H} has $f_2^{\mathrm{H}}(1) = 0$, $f_2^{\mathrm{H}}(2) = 1$, or vice versa, and is optimal. Assume the Huffman code f_{m-1}^{H} is optimal for I_{m-1}, whatever the probability distribution. Suppose further that