CHAPTER 1

Entropy and lossless source coding

1.1. Introduction and motivations

When trying to compress files on a computer system, the following question naturally arises: if we had the best possible compression program, how large would be our compressed file system? What is the ultimate achievable compression ratio for our file system? As a matter of fact, we are not really interested in the file system residing on a peculiar computer, but in any reasonable collection of files that could appear on private computer, or on a company network. In that form, this is not a mathematical question, since we do not have any definition of what could be a reasonable collection of files. It is hard to define an optimal solution to a problem that has not even be stated.

It is just possible to observe that `gzip` usually outperforms `compress`, and that in turn `bzip2` usually outperforms `gzip`. This is illustrated in the following table:

Method File	`bzip2 -1`	`bzip2 -9`	`gzip -1`	`gzip -9`	`compress`	originalsize
lecture.lyx	30705	28747	41203	33629		135114
gzip.man	5131	5131	6063	5411	7033	13923

As the original question concerning the performance of data compression systems is an empirical one, many efforts have been dedicated to build relevant collections of benchmarks or corpus to compare the performance of compression systems on realistic situations. Those corpus are available through the Web (see [**?**]).

In order to develop a theory of data compression, C.E. Shannon suggested to model text production as a random phenomenon. This bet has been surprisingly efficient and fertile.

1.2. Notations

In the sequel, $\mathcal{X}$ denotes a set called the *alphabet*. In chapters dedicated to lossless (text) coding the alphabet is assumed to be finite. The elements of $\mathcal{X}$ are called *letters* or *symbols*. The ASCII alphabet, the Greek alphhabet provide examples of alphabets. Sequence of 80 ASCII characters (lines of FORTRAN programs) provide another example of alphabet. The binary alphabet is $\{0,1\}$, its symbols are called bits. The notation x_m^n is a shorthand for $x_m, \ldots, x_n$.

$\mathcal{X}^*$ denotes the set of finite *words* (finite sequences of letters). The *length* of a word x is just the number of letters that constitute it. It is denoted by $\ell(x)$. A word x is a *prefix* of a word y if there exists another word (possibly empty) z such that $y = xz$.

DEFINITION 1.2.1. [CODES & CODEBOOKS] Let $\mathcal{Y}$ denote another alphabet. A (lossless) *code* is a (one-to-one) mapping from $\mathcal{X}^*$ on $\mathcal{Y}^*$. If the mapping is denoted by f, the range of $\mathcal{X}^*$ under f is called the *codebook*. The decoding function is denoted by ϕ. $\phi \circ f = \mathrm{Id}$.

The encoding alphabet $\mathcal{Y}$ will usually be the binary alphabet $\{0,1\}$.

DEFINITION 1.2.2. [COMPRESSION RATE] The *compression rate* of f on x is $\ell(f(x))/\ell(x)$. $f(x)$ is the *codeword* associated with x.

Whatever the alphabet$\mathcal{X}$ is, for each integer$n$, $\mathcal{X}^n$ is an obvious measurable space, and there is no difficulty in defining probability distributions on this finite set. A probability distribution or law on $\mathcal{X}^n$ is completely defined by a mapping P^n from $\mathcal{X}^n$ on $[0,1]$ such that

$$\sum_{w \in \mathcal{X}^n} P^n(w) = 1 \,.$$

The probability of $A \subseteq \mathcal{X}^n$is

$$P^n\{A\} = \sum_{w \in A} P^n(w) \quad .$$

DEFINITION 1.2.3. [CONSISTENT FAMILY OF PROBABILITY DISTRIBUTIONS] A sequence of probability distributions $(P^n)_{n \in \mathbb{N}}$ over $\mathcal{X}^n$ is said to be consistent if for all $n, m \in \mathbb{N}$,for all $A \subset \mathcal{X}^n$

$$P^{n+m}\{A \times \mathcal{X}^m\} = P^n\{A\} \ .$$

Great deal of works can be done by just considering consistent sequences of probability distributions. It is nevertheless useful and sometimes mandatory to observe that by the *Kolmogorov consistency theorem*, there exists a unique non-negative function $\mathbb{P}$ on the σ-algebra $\mathcal{F}$ of subsets of $\mathcal{X}^{\mathbb{N}}$ generated by of

countably many unions, intersections and complementations of sets of the form $A \times \mathcal{X}^{\mathbb{N}}$ where $A \subseteq \mathcal{X}^n$for some $n \in \mathbb{N}$ (those sets are called cylinders, and $\mathcal{F}$ is called the cylindrical σ-algebra) such that

(1) if $A_1, \ldots A_m, \ldots$ are disjoint sets from $\mathcal{F}$,

$$\mathbb{P}\left\{\cup_{i=1}^{\infty} A_i\right\} = \sum_{i=1}^{\infty} \mathbb{P}\left\{A_i\right\} \ ,$$

(2) $\mathbb{P}\left\{\mathcal{X}^{\mathbb{N}}\right\} = 1$,
(3) For all $A \subseteq \mathcal{X}^n$,$\mathbb{P}\left\{A \times \mathcal{X}^{\mathbb{N}}\right\} = \mathbb{P}\left\{A\right\} = P^n\left\{A\right\}$.

Little more works ensures that the set $\mathcal{X}_{-\infty}^{\infty} = \mathcal{X}_{-\mathbb{N}}^{\mathbb{N}}$ of doubly infinite sequences of letters is also a measurable space. It will be equipped with the cylindrical σ-algebra: i.e. the σ-algebra defined by conditions on finitely many coordinates.

DEFINITION 1.2.4. [SOURCE] Any consistent family of probability distributions $\left(P_{-m}^{n}\right)$on $\mathcal{X}_{-m}^{n}$, or equivalently any probability on $\mathcal{X}_{-\infty}^{\infty}$ defines a source on alphabet $\mathcal{X}$.

1.3. Operational definition of entropy

Here, we will adopt the information theoretical approach. Texts are the output of sources. A *source* is nothing but a probability distribution defined on $\mathcal{X}_{-\infty}^{\infty}$ (with respect to the cylindrical σ-algebra). A source defines an $\mathcal{X}$-valued process. It is completely determined by its finite-dimensional projections, i.e. by the definition of

$$\mathbb{P}\{X_m^n \in A\} \quad \text{for all}, m, n, \quad A \subseteq \mathcal{X}^{n-m+1} \ .$$

A source is *memoryless* if the distribution of X_{n+1} does not depend on $X_{-\infty}^{n}$. A source is *stationary* if and only if for all $m \in \mathbb{Z}$, $(X_.)$ and $(X_{.+m})$ have the same distribution.

DEFINITION 1.3.1. [COMPRESSION RATE OVER WORDS OF LENGTH n] If (f, ϕ) denotes a pair of encoding/decoding functions, the compression rate of f on words of length n is:

$$\mathbb{E}\left[\frac{\ell(f(X_1^n))}{n}\right] .$$

The compression rate R is said to be achievable iff there exists a code (f, ϕ) such that

$$\limsup_n \mathbb{E}\left[\frac{\ell(f(X_1^n))}{n}\right] \leq \mathsf{R}.$$

The ultimate compression rate we are looking for is the infimum of all achievable compression rates.

Note that the ultimate compression rate is defined operationally by asymptotical conditions. This is a trademark of information theoretical approaches: First define an asymptotical quantity that reflects the engineering limits under consideration; then investigate the speed at which the asymptote is approached. Information Theory sets theoretical limits to the performance of coding techniques. But it usually does not propose efficient techniques that can approach those theoretical limits, it sometimes gives hints.

1.4. Uniquely-decodable codes

For practical purposes, and for many theoretical purposes as well, it is not enough to ask a lossless code to be one-to-one. We also would like it to be uniquely decodable.

DEFINITION 1.4.1. [UNIQUELY DECODABLE CODES] A code (f, ϕ) is *uniquely decodable*, iff for any collection $w_1, w_2 \dots w_n$ of words, there is no collection $w'_1 \dots w'_m$ of words such that:

$$f(w_1)f(w_2)\dots f(w_n) = f(w'_1)f(w'_2)\dots f(w'_m) \ ,$$

with $(w_1, w_2, \dots w_n) \neq (w'_1, \dots w'_m)$.

If a code is uniquely-decodable, then from the concatenation of codewords, it is still possible to recover the original collection of words.

CLAIM 1.4.2. A uniquely decodable code defines a one-to-one mapping from $(\mathcal{X}^*)^*$ on $\mathcal{Y}^*$.

If no codeword is a prefix of another codeword, then the code is uniquely decodable.

DEFINITION 1.4.3. [PREFIX CODES]A set of words $\mathcal{A} \subseteq \mathcal{X}^*$ is *prefix* if no word in $\mathcal{A}$ is a prefix of another word in $\mathcal{A}$.

A very desirable property of a code: instantaneously decodability. A code is instantaneous, if end of codewords can be detected without looking to the future.

CLAIM 1.4.4. A code is instantaneously decodable if the codebook $f(\mathcal{X}^*)$ is prefix.

1.5. Kraft-McMillan inequalities

Kraft-Mac-Millan inequalities relate length of codewords and probabilities, they constitute the bridge between source coding and statistics.

THEOREM 1.5.1. [KRAFT MCMILLAN INEQUALITY] *Let (f, ϕ) denote a uniquely-decodable code from $\mathcal{X}^*$ on $\mathcal{Y}^*$, then:*

$$\sum_{w \in \mathcal{X}^*} |\mathcal{Y}|^{-\ell(f[w])} \leq 1 \ .$$

The core of the argument boils down to the following observation: if $s > 1$ then s^n grows faster than any polynomial with respect to n.

PROOF. Note that it is enough to check that for all m:

$$\sum_{w \in \mathcal{X}^*, \ell(w) \leq m} |\mathcal{Y}|^{-\ell(f([w])} \leq 1.$$

Fix $m \in \mathbb{N}$, let $p \stackrel{\Delta}{=} \max_{w \in \mathcal{X}^*, \ell(w) \leq m} \ell(f[w])$.

$$\begin{aligned}
\left(\sum_{\ell(w) \leq m} |\mathcal{Y}|^{-\ell(f[w])} \right)^n &\stackrel{(a)}{=} \sum_{w_1 \ldots w_n, \ell(w_i) \leq m} |\mathcal{Y}|^{-\sum_{i \leq n} \ell(f[w_i])} \\
&\stackrel{(b)}{=} \sum_{l \leq np} \sum_{w_1, w_n : \sum_i \ell(f[w_i]) = l} |\mathcal{Y}|^{-l} \\
&\stackrel{(c)}{\leq} \sum_{l \leq np} |\mathcal{Y}|^l |\mathcal{Y}|^{-l} \\
&\stackrel{(d)}{\leq} np \ ,
\end{aligned}$$

where (a) follows from power expansion, (b) follows from reordering the finite sum, (c) follows from the unique decodability of the code, any word of length l on $\mathcal{Y}$ can be parsed into at most one collection of codewords, and there are $|\mathcal{Y}|^l$ words of length l.This establishes that

$$\sum_{\ell(w) \leq m} |\mathcal{Y}|^{-\ell(f[w])} \leq 1.$$

□

REMARK. The Kraft-MacMillan inequality points out the following correspondence between uniquely decodable codes and probabilities on $\mathcal{X}^*$. Let Q be defined on $\mathcal{X}^* \cup \{\perp\}$ by

$$Q(w) \triangleq |\mathcal{Y}|^{-\ell(f[w])}$$

for $w \in \mathcal{X}^*$, and $Q(\perp) \triangleq 1 - \sum_{w \in \mathcal{X}^*} Q(w)$.

The relative entropy between two probability distributions will be a cornerstone of Information Theory (an of many other areas of probability and statistics). The Kraft-McMillan inequality will be useful in connection with the notion of relative entropy.

1.6. Relative entropy

DEFINITION 1.6.1. [RELATIVE ENTROPY] The *relative entropy* between probability distributions P and Q on the countable set E is defined by:

$$D(P \,\|\, Q) \triangleq \begin{cases} \mathbb{E}_P \left[\log \frac{P(w)}{Q(w)}\right] & \text{if } P(w) > 0 \Rightarrow Q(w) > 0, \\ \infty & \text{otherwise.} \end{cases}$$

Recall that a function f on a vector space F is convex if and only if for all $x, y \in F$, for all $\lambda \in [0,1]$:

$$\lambda f(x) + (1-\lambda) f(y) \geq f\left(\lambda x + (1-\lambda) y\right).$$

By induction on the number of summands, it follows:

THEOREM 1.6.2. [JENSEN INEQUALITY] *If f is a convex function on $\mathbb{R}$,and P denote a probability distribution on a discrete subset of $\mathbb{R}$, then*

$$\mathbb{E}_P\left[f(X)\right] \geq f\left(\mathbb{E}\left[X\right]\right) .$$

The theorem actually holds for all probability distributions on $\mathbb{R}$and convex functions.

The following proposition is a direct consequence of the strict concavity of logarithms and Jensen inequality:

PROPOSITION 1.6.3. *$D(P \,\|\, Q) \geq 0$, and equality is only possible when $P = Q$.*

PROOF. There is nothing to prove if there exists some x such that $P(x) = 0$ and $Q(x) > 0$. Let P and Qdenote two probability distributions such that $P(x) > 0 \Rightarrow Q(x) > 0$.

$$\begin{aligned} D(P \,\|\, Q) &= \sum_x P(x) \log \frac{P(x)}{Q(x)} \\ &= \sum_{x:P(x)>0} P(x) \left(-\log \frac{Q(x)}{P(x)} \right) \\ &\geq -\log \left(\sum_{x:P(x)>0} P(x) \left(\frac{Q(x)}{P(x)} \right) \right) \\ &= -\log \left(\sum_{x:P(x)>0} Q(x) \right) \\ &\geq -\log(1) \ . \end{aligned}$$

□

EXERCISE 1.7. [LOG-SUM INEQUALITY] Check the following inequality :

$$\sum_{i=1}^{n} a_i \log \frac{a_i}{b_i} \geq \left(\sum_{i=1}^{n} a_i \right) \log \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \, .$$

EXERCISE 1.8. [CONVEXITY OF RELATIVE ENTROPY] Derive from the preceding inequality that relative entropy is convex with respect to its two arguments.

The Kraft-McMillan inequality and the positivity of the relative entropy enable to deduce a lower bound on the compression rate.

1.9. A lower bound on compression rate

THEOREM 1.9.1. [CONVERSE LOSLESS CODING THEOREM] *Let P_n denote the probability distribution on $\mathcal{X}^n$ defined by a source, then for any uniquely decodable code $(f, \phi)$with coding alphabet $\mathcal{Y}$:*

$$\mathbb{E}_{P_n} \left[\ell(f[w]) \right] \geq \frac{1}{\log |\mathcal{Y}|} \mathbb{E}_{P_n} \left[-\log P_n(w) \right] \ . \tag{1.9.1}$$

PROOF. From the Kraft-McMillan inequality it is immediate that

$$\sum_{w\in\mathcal{X}^*} |\mathcal{Y}|^{-\ell(f[w])} \leq 1.$$

Hence it is possible to define a probability Q_n on $\mathcal{X}^n \cup \{\perp\}$ with $Q_n(x_1^n) \triangleq |\mathcal{Y}|^{-\ell(f[x_1^n])}$. Moreover

$$\begin{aligned} D(P_n \,\|\, Q_n) &\overset{(a)}{=} \mathbb{E}_{P_n}\left[\log P_n(X_1^n)\right] - \mathbb{E}_{P_n}\left[-\log Q_n(X_1^n)\right] \\ &\overset{(b)}{=} \mathbb{E}_{P_n}\left[\log P_n(X_1^n)\right] + \mathbb{E}_{P_n}\left[\log|\mathcal{Y}|\,\ell(f[X_1^n])\right]. \end{aligned}$$

Now the positivity of relative entropy implies the theorem. □

1.10. Huffman coding and entropy

1.10.1. Building an optimal prefix code. Soon after Shannon established the noiseless source coding theorem (1948), Huffman proposed a technique to design instantaneously decodable (prefix) codes for finite sources, i.e. probabilities on finite subsets of $\mathcal{X}^*$. This method is interesting for several reasons: it is simple, it raises interesting algorithmic issues, and moreover it is useful since it is still used in data compression standards such as JPEG or MPEG.

1.10.1.1. *A simple and useful property of some optimal prefix codes.* For the sake of brevity , we assume that $\mathcal{Y} = \{0, 1\}$.

Assume $\mathcal{A} \subset \mathcal{X}^*$ is finite and provided with probability P. There exists a code with minimal average length, because of the finiteness of the number of possible length vectors smaller than $|\mathcal{A}|$.

PROPOSITION 1.10.1. *There exists a prefix code (f, ϕ) with minimal average length such that the following two properties hold:*

(1) *If w_1 and w_2 are the least frequent words in $\mathcal{A}$, then $\ell(f[w_1]) = \ell(f[w_2])$ and $f[w_1]$ and $f([w_2])$ differ only in the last symbol.*
(2) *If $P\{w_1\} \leq P\{w_2\}$ then $\ell(f[w_1]) \geq \ell(f[w_2])$.*

Note that those two properties imply that the two longest codewords have the same length.

PROOF. The proof starts with an optimal prefix code (f, ϕ), and proceeds by rearrangement: if (f, ϕ) satisfies the two properties, we are done.

Assume the second property is violated by w_1 and w_2, then by exchanging the roles of $f[w_1]$ and $f[w_2]$, we get another prefix code with strictly smaller average length, contradicting the optimality assumption.

Assume the first property is violated, let w_1 denote the least frequent symbol and w_2 the second least frequent symbol. If there exists w_3 such that $f[w_1]$ and $f([w_3])$ differ only in the last symbol, then as $\ell(f[w_1])$ is maximal (property 2), $\ell(f[w_3]) \geq \ell(f[w_2])$ and we may exchange the roles of $f[w_2]$ and $f([w_3])$ without destroying either the prefix property or the optimality (hence property 2), and then property 1 is satisfied. If there is no w_3 such that $f[w_3]$ and $f[w_1]$ differ only in the last symbol, then we can get rid of the last symbol in $f[w_1]$ and get an shorter prefix code, contradicting the optimality assumption □

1.10.1.2. *Huffman's construction.* To describe Huffman's construction, it is profitable to use the correspondence between prefix sets over $\{0,1\}$ and labelled (planar) binary trees.

A priority queue is an *abstract data type* parametrized by a set E and a priority function key

such for any (possibly empty)collection of pairs $(x_i, \mathsf{key}(x_i))$, one may build a priority queue PQ such that:

(1) each item in PQ is associated with exactly one pair $(x_i, \mathsf{key}(x_i)$,
(2) one may test whether the queue is empty or not,
(3) one may extract the item $(x_m, \mathsf{key}(x_m))$ with minimal key, obtaining on one hand, $(x_m, \mathsf{key}(x_m))$ and a priority queue containing all other items,
(4) one may update the priority of an item in the priority queue.

Huffmann construction is described as follows:

(1) Build a priority queue PQ associated with all pairs $(\langle w_i, \epsilon\rangle, P\{w_i\})$
(2) While PQ contains at least two items do the following steps:

- extract the item with minimal priority (T_i, P_i) and let $\langle T_j, P_j\rangle$ denote the item it with minimal priority in the resulting priority queue.
- Replace T_j by $((T_i, 1), (T_j, 0))$ in the item it , and replace priority P_j by $P_j + P_i$ in this item, update the priority queue.

(1) Output the code defined by the tree T in the information field of the (unique) remaining item.

THEOREM 1.10.2. [Huffmann code optimality] *Huffman's codes have optimal average length among prefix codes..*

PROOF. The proof proceeds by induction on the number of words that have to be encoded. If there are two words, the construction is trivially optimal. Now assume that it is optimal when there are less than n words, and that $|\mathcal{A}| = n+1$.

Let w_n and w_{n+1} denote the two least frequent symbols. $f[w_n]$ and and $f[w_{n+1}]$ have equal length and they differ only in the last symbol. $\mathcal{A}'$ is defined by $\mathcal{A}' = \mathcal{A} \setminus \{w_n, w_{n+1}\} \cup \{w_{n+2}\}$, and equipped with a probability distribution P' defined by $P'\{w_j\} = P\{w_j\}$ for $j < n$ and $P'\{w_{n+2}\} = P\{w_n\} + P\{w_{n+1}\}$. The code produced by Huffman's construction on $\mathcal{A}'$ assigns codeword $f[w_j]$ to any w_j with $j < n$ and all but the last symbols of $f[w_n]$ to w_{n+2}. It is a prefix code, by the induction hypothesis, it has optimal length. Assume now that f' defines an optimal prefix code satisfying properties ... and ..., f' defines a code on $\mathcal{A}'$ in the same way. Let us compare the average length of those two codes:

$$\begin{aligned}
\mathbb{E}_{P'}[\ell(f[w])] &\overset{(a)}{\leq} \mathbb{E}_{P'}[\ell(f'[w])] \\
&\overset{(b)}{\leq} \mathbb{E}_{P}[\ell(f'[w])] - P\{w_n w_{n+1}\} \\
&\overset{(c)}{\leq} \mathbb{E}_{P}[\ell(f[w])] - P\{w_n w_{n+1}\}
\end{aligned}$$

On the other hand

$$\mathbb{E}_{P'}[\ell(f[w])] = \mathbb{E}_{P}[\ell(f[w])] - P\{w_n w_{n+1}\},$$

thus

$$\mathbb{E}_{P}[\ell(f[w])] = \mathbb{E}_{P}[\ell(f'[w])] \ .$$

□

1.11. The converse to the Kraft-McMillan inequality

The optimality of Huffmann codes among prefix codes leaves open the following questions:

(1) Are prefix codes optimal among uniquely decodable codes ?
(2) How close is the infimum over all achievable compression rates to the entropy rate ?

THEOREM 1.11.1. [CONVERSE KRAFT-MILLAN INEQUALITY] *If* $\lambda : \mathcal{A} \subseteq \mathcal{X}^* \to \mathbb{N}$ *satisfies*

$$\sum_{w \in \mathcal{A}} |\mathcal{Y}|^{-\lambda(w)} \leq 1,$$

then there exists a prefix code (f, ϕ) *on alphabet* $\mathcal{Y}$*, such that* $\ell(f[w]) = \lambda(w)$.

PROOF. To alleviate notations, let us assume that $\mathcal{Y} = \{0, 1\}$. Without loss of generality, assume $\mathcal{A} = \{w_1, \dots w_m\}$, and that $\lambda(w_i) < \lambda(w_j) \Rightarrow i < j$. Let the encoding of w_i be denoted by $f(w_i)$ and be defined as the $\lambda(w_i)$ first bits of the binary expansion of

$$\sum_{j<i} 2^{-\lambda(w_j)} + 2^{-\lambda(w_i)-1},$$

which also coincides with the binary expansion of:

$$\sum_{j<i} 2^{-\lambda(w_j)}.$$

It is enough to check that this defines a prefix-free code. Assume on the contrary that $f(w_i)$ is a prefix of $f(w_{i'})$ while $i \neq i'$. This entails $i < i'$. Then we have:

$$\sum_{i \leq j < i'} 2^{-\lambda(w_j)} < 2^{-\lambda(w_i)},$$

which is impossible. □

COROLLARY 1.11.2. [DIRECT LOSLESS CODING THEOREM] *For any $\mathcal{A} \subseteq \mathcal{X}^*$, provided with a probability distribution P, there exists a prefix code (f, ϕ), such that $\ell(f[w]) = \lceil -\log P\{w\} \rceil$, moreover:*

$$\mathbb{E}[\ell(f[w])] \leq H(P) + 1 \ .$$

How effective is the converse to the Kraft-MacMillan Theorem ? Provide an efficient coding-decoding procedeure.

1.12. Arithmetic coding

1.12.1. Shannon-Fano-Elias codes. If handling long blocks is not an issue, Huffman coding is quite satisfactory: the redundancy of Huffman codes is upper bounded by $1/n$, coding consists in table lookup and decoding in traversing a tree. But the size of coding table grows exponentially with n. Hence an algebraic improvement of redundancy costs an exponential amount of space. This is one reason to look at other lossless coding techniques. This first technique to be exposed has not been the most useful in practical implementations till now, but it has a tight connection with statistical modeling and has had an important impact in non-parametric statistics, it is called *arithmetic coding*. Arithmetic coding stems from an ancient idea developed by Shannon and Fano at the very beginning of Information Theory.

1.12.2. Coding a single symbol. Let $\mathcal{A} = \mathcal{X}$. Let $P\{w\}$ denote $P\{X_1 = w\}$ in this paragraph. Remember that from the converse of the Kraft-McMillan inequality, as $\sum_{w\in\mathcal{X}} 2^{-\lceil -\log P\{w\}\rceil -1} \leq 1$, there exists a prefix code f, ϕ that assign to each symbol x a code-length equal to $\lceil -\log P\{x\}\rceil + 1$.

Such a code can be designed in the following way. Choose an ordering on $\mathcal{A} = \mathcal{X}$. Define low and up from as:

$$\begin{aligned} \text{low}(x) &\triangleq \sum_{x'<x} P\{x'\} \\ \text{up}(x) &\triangleq \sum_{x'\leq x} P\{x'\}\,. \end{aligned}$$

Define $f(x)$ as the

$$1 + \lceil -\log P\{x\}\rceil$$

bits of the binary expansion of

$$\frac{1}{2}[\text{low }(x) + \text{up}(x)] = \text{low}(x) + \frac{P\{x\}}{2}.$$

Let us check that this defines a proper prefix code. Assume that $f(x)$ is a prefix of $f(x')$and $(x < x')$, this implies that the binary expansions of low $(x) + \frac{1}{2}P\{x\}$ and $\text{low}(x') + \frac{1}{2}P\{x'\}$ do not differ in their first $1 + \lceil -\log P\{x\}\rceil$ positions. i.e. they differ by less than

$$2^{-1-\lceil -\log P\{x\}\rceil} \leq \frac{P\{x\}}{2},$$

but

$$\text{low}(x') + \frac{1}{2}P\{x'\} - \text{low}(x) - \frac{1}{2}P\{x\} > \frac{P\{x\}}{2}\,.$$

The case $x > x'$ is treated in a similar way (or we may impose that the letters are ordered in a non-increasing manner according to their probabilities).

1.12.3. Coding a sequence of symbols. This construction provides with a prefix code for $\mathcal{X}$ where each codeword has nearly ideal code-length :

$$-\log P\{x\}.$$

As such it does not seem to provide an improvement with respect to Huffman codes. The main merit of the Shannon-Fano-Elias coding method is to be graciously extendible to the coding of sequences of symbols.

Let us consider the encoding of $\mathcal{X}^n$. The order on $\mathcal{X}$ induces a unique lexicographic order on $\mathcal{X}^n$. And $x_1^n \in \mathcal{X}^n$ can be encoded by taking the first $1 + \lceil -\log P^n\{x_1^n\}\rceil$ bits in the binary expansion of

$$P^n\{X_1^n < x_1^n\} + \frac{1}{2}P^n\{x_1^n\}.$$

The most appealing aspect of this encoding is that any good procedure to compute $P^n\{X_1^n < x_1^n\}$ from $P^n\{X_1^{n-1} < x_1^{n-1}\}$ results in a good procedure to encode x_1^n. As a matter of fact, define by extension for any $x_1^n \in \mathcal{X}^n$:

$$\text{low } (x_1^n) \stackrel{\Delta}{=} P^n\{X_1^n < x_1^n\} \quad \text{up}(x_1^n) \stackrel{\Delta}{=} P^n\{X_1^n \leq x_1^n\}$$

$\langle \text{low}(x_1^n), {}^(x_1^n)\rangle$ is commonly called the tag associated with x_1^n. let us also define $\text{width}(x_1^n) \stackrel{\Delta}{=} {}^(x_1^n) - \text{low}(x_1^n)$. $\text{low}(x_1^n)$ and ${}^(x_1^n)$ are computed in the following way:

$$\begin{aligned}
\text{low}(x_1^n) &= \text{low}(x_1^{n-1}) + \left[\text{width}(x_1^{n-1})\right] P\left\{X_n < x_n \mid X_1^{n-1} = x_1^{n-1}\right\} \\
\text{up}(x_1^n) &= \text{low}(x_1^{n-1}) + \left[\text{width}(x_1^{n-1})\right] P\left\{X_n \leq x_n \mid X_1^{n-1} = x_1^{n-1}\right\} \\
\text{width}(x_1^n) &= \text{width}(x_1^{n-1}) P^n\left\{X_n = x_n \mid X_1^{n-1} = x_1^{n-1}\right\}.
\end{aligned}$$

The tag could be equivalently chosen as $\langle \text{low}(x_1^n), \text{width}(x_1^n)\rangle$. Once the tag is known, encoding is straightforward. Shannon-Fano-Elias coding coding does not require exponentially large tables and its redundancy per symbol is less than $2/n$. Nevertheless this is not enough to make it practical.

1.12.4. Pure arithmetic coding. Here is list of problems that have to be fixed in order to make Shannon-Fano-Elias coding a practical idea:

(1) coding has to be performed in an incremental manner. It is desirable to be able to output the codeword beginning before having totally processed the input word.
(2) tag computations should be simple. The Shannon-Fano-Elias method relies on exact arithmetic, and multiplications. It is desirable to used fixed precision arithmetic and to avoid multiplications.

Those two problems have been (at least partially) fixed in the late seventies.

Pure arithmetic coding deals with the first question and leaves aside the implementational issues raised by exact arithmetic. From the definition of the Shannon-Fano-Elias procedure, it follows that the sequence of intervals $[\text{low}(x_1^k), {}^(x_1^k))$ is decreasing. We will first describe a simple set of rules that will replace this sequence of intervals by a sequence of intervals which width remains bounded from below by $1/4$.

Assume that $k < n$ and $[\text{low}(x_1^k), {}^(x_1^k)) \subseteq [0, 1/2)$ then as $[\text{low}(x_1^{k+1}), {}^(x_1^{k+1})) \subseteq [\text{low}(x_1^k), {}^(x_1^k))$, we already know that the first bit of the codeword associated with x_1^n will be 0. We could thus replace $[\text{low}(x_1^k), {}^(x_1^k))$ by $[2 \times \text{low}(x_1^k), 2 \times {}^(x_1^k))$ and immediately output 0.

In an analogous way, if $[\text{low}(x_1^k), {}^(x_1^k)) \subseteq [1/2, 1)$ then it is replaced by $[2 * (\text{low}(x_1^k) - \frac{1}{2}), 2 * ({}^(x_1^k) - \frac{1}{2}))$ and 1 is immediately output.

The third rule is more subtle. The first two rules do not prevent $[\mathrm{low}(x_1^k), \mathrm{high}(x_1^k))$ to shrink around $1/2$ like $[1/2 - 1/2^k, 1/2 + 1/2^k)$. In such a case, we are not in a position to decide whether the first codeword symbol will be 1 or 0 until the last input symbol is read. Nevertheless, if $[\mathrm{low}(x_1^k), \mathrm{high}(x_1^k)) \subseteq [\frac{1}{4}, \frac{3}{4})$, we already know that the first and the second bit of the codeword will be different. This prompts us to follow the following rule: replace $[\mathrm{low}(x_1^k), \mathrm{high}(x_1^k))$ by $[2 \times (\mathrm{low}(x_1^k) - \frac{1}{4}), 2 \times (\mathrm{high}(x_1^k) - \frac{1}{4}))$, and note that the first and second bits of the codeword will be different.

Beware that the three rules should be used in an iterative manner. This is packaged in the following algorithm.