Information Theory

Xavier Servot *

Février 2019

1 Discrete Probabilities

The *sample space* Ω is a finite or countably infinite set. An *event* is any subset of the sample space, and is thus an element of $\mathcal{P}(\Omega)$. The *impossible event* is $\varnothing$ and the *certain event* is Ω.

Let the *probability measure* be a function

$$P\colon \mathcal{P}(\Omega) \to [0;1]$$

which verify the properties

- $P(\Omega) = 1$
- $\forall (E_i)_{i=1}^{\infty} \in \mathcal{P}(\Omega)^{\mathbb{N}}$ mutually exclusive, $P(\bigcap_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} P(E_i)$
  Where mutually exclusive means that $\forall i \in \mathbb{N}, \forall j \in \mathbb{N} \setminus \{i\}, E_i \cap E_j = \varnothing$

Let E be a finite or countably finite set. A *discrete random varible* is a function

$$X\colon \Omega \to E$$

The *probability distribution* of a random variable X is a function $P_X\colon X(\Omega) \to [0;1]$ such that

$$P_X(x) = P(\{\omega \in \Omega \mid X(\omega) = x\})$$

It is not a probability measure since the probability distribution's domain is $X(\Omega)$ and not $\mathcal{P}(X(\Omega))$.

*EPFL

For now, let's note $\{\omega \in \Omega \,|\, X(\omega) = x\}$ by $\{X = x\}$. The property above then becomes

$$P_X(x) = P(X = x)$$

If $(X_i)_{i=1}^n$ are random variables on Ω, let their *joint probability distribution* be a function $P_{X_1 \dots X_n} \colon X_1(\Omega) \times \cdots \times X_n(\Omega) \to [0; 1]$ such that

$$P_{X_1 \dots X_n}(x_1, \dots, x_n) = P(\bigcap_{i=1}^{n} \{X_i = x_i\})$$

Which verifies (from the second property of a probability measure)

$$\sum_{x_i \in X_i(\Omega)} P_{X_1 \dots X_n}(x_1, \dots, x_n) = P_{X_1 \dots X_{i-1} X_{i+1} \dots X_n}(x_1, \dots, x_{i-1} x_{i+1} \dots x_n)$$

The random varibles $(X_i)_{i=1}^n$ are *statistically independant* when $\forall (x_1, \dots, x_n) \in X_1(\Omega) \times \cdots \times X_n(\Omega)$

$$P_{X_1 \dots X_n}(x_1, \dots, x_n) = \prod_{i=1}^{n} P_{X_i}(x_i)$$

Let F be a function $F \colon X(\Omega) \to \mathbb{R}$. Then its *expected value* $\mathrm{E}[F(X)]$ is the real number

$$\mathrm{E}[F(X)] = \sum_{x \in X(\Omega)} F(x) P_X(x)$$

We also consider the *conditional probability distribution*: given two random variable X and Y on Ω, $\forall (x, y) \in X(\Omega) \times Y(\Omega)$ let

$$P_{Y|X}(y|x) = \frac{P_{XY}(x, y)}{P_X(x)}$$

Now let $X = \mathrm{Id}_\Omega$ be the random variable $X \colon \omega \in \Omega \mapsto \omega \in \Omega$. Then the definition of the conditional probability distribution can be adapted to a probability measure P: we have $P_X = P$ and $\forall (A, B) \in \mathcal{P}(\Omega)^2$,

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

2 Measuring the information

2.1 Hartley's measure

Consider that a symbol can take D possible values. Then n of those symbols can take D^n possible values. Now the information conveyed by n symbols ought to be n times as much as for a single symbol. So taking the log:

$$\log(D^n) = n \log(D)$$

we'd have a measure of information that verifies our rule. Furthemore the base selected for the logarithm fixes the size of the unit of information.

More formally, let X be a real random variable. The measure of information of that variable is

$$I(X) = \log(\operatorname{Card} \operatorname{Im}(X))$$

where $\operatorname{Card} \operatorname{Im}(X)$ is the number of possible values of X.

However, Hartley's measure of information has its limits. Consider the experiment where we have

- One box with two white balls and two black balls. (1)
- One box with one white balls and three black balls. (2)

Formally the experiments (1 or 2) are to choose randomly a ball from the box described. Let X be the random variable such that choosing a white ball maps to 0 and choosing the black maps to 1. We have that $\operatorname{Card} \operatorname{Im}(X) = 2$ thus we can encode the information using one bits.

Yet in the experiment (2) we *expect* from the random variable to be 0 more than 1. Thus the information of a black ball being picked should be stored using less space because choosing a black ball was already most expected in the first place.

Hartley's measure does not take advantage of the probability distribution to encode the information.

2.2 Shannon's measure

Shannon's measure relies heavily on Hartley's measure. The intuition behind it goes as follows:

In the experiment (2) we had 1 chance out of 4 to choose a white ball. Thus we could say that the white ball was only 1 possibility out of 4, and the measure of information for this case would be

$$\log_2(4) = 2$$

For the black balls, we had 3 chances out of 4 to choose a white ball. Thus we could say that the black black ball was to be picked amongst 4/3 items (whatever that may mean), and the measure would be

$$\log_2(4/3)$$

We could reconciliate that fact by measuring the information with the probabilities weighed in:

$$1/4 \log_2(4) + 3/4 \log_2(4/3)$$

More formally, let X be a real random variable and b the length of a base unit of information. Let $\operatorname{supp} P_X = \{x \in X \mid P_X(x) > 0\}$ since log is not defined at 0. Considering that $\log_b(1/x) = -\log_b(x)$, the *uncertainty* of X is

$$H(X) := -\sum_{x \in \operatorname{supp} P_X} P_X(x) \log_b(P_X(x))$$

Now we have not talked about *information* but *uncertainty* in that definition. Shannon always defined information to be the difference in uncertainties. Thus $H(X)$ can be called information since when observing X, we can be certain of its value thus the uncertainty drops to 0, yet the uncertainty of its value before that was $H(X)$ (by definition).

We can think of computing H by taking the average of the values of $\log_b(P_X)$ and we in fact have

$$H(X) = \mathrm{E}[-\log_b(P_X)]$$

There is a link between Hartley's measure of information and Shannon's uncertainty. Let $\mathcal{A}$ be a finite alphabet. Consider the experiment of choosing at random and with equal probability an element of that alphabet. Then $\forall x \in X(\mathcal{A}), P_X(x) = \frac{1}{\operatorname{Card} \mathcal{A}}$. In particular:

$$H(X) = \sum_{x \in \operatorname{supp} X} \frac{1}{\operatorname{Card} \mathcal{A}} \log_b(\operatorname{Card} \mathcal{A})$$

$$= \log_b(\operatorname{Card} \mathcal{A})$$
$$= I(X)$$

Thus when the probability is constant, $H(X) = I(X)$.

Proposition 2.1 (IT-inequality)**.**

$$\forall r > 0, \ln(r) \leq r - 1$$

Proof. Multiple proofs are given

- $x \mapsto x - 1$ is the tengent of ln at 1. Now ln is concave i.e. its curve is under its tengents and we thus have the inequality.
- Let $f : x \in \mathbb{R}_{\geq 0} \mapsto \ln x - x + 1$. Let's find the maximum of f.

  $f'(x) = 0$ iff. $x = 1$. Also $\lim_{x \to 0^+} f(x) = -\infty$ and $\lim_{x \to +\infty} = -\infty$.

  Also $f'' \leq 0$ which means it is a maximum in 1. Now $f(1) = 0$ thus

  $$\forall x \in \mathbb{R}_{>0}, \ln x \leq x - 1$$

- We first prove that $\forall x > -1$, $e^x \geq x + 1$. We'd then have for $r = x + 1$ (and thus $r > 0$),

  $$\ln(e^x) \geq \ln(x + 1) \Leftrightarrow r - 1 \geq \ln(r)$$

  The proof is immediate if we define $e^{\cdot}$ as a power series.

  Let's use $e^x := \lim_{n \to +\infty}(1 + \frac{x}{n})^n$ instead. We have in particular that $\forall n \in \mathbb{N}$, using Binomial theorem,

  $$(1 + \frac{x}{n})^n \leq 1 + n\frac{x}{n} = 1 + x$$

□

Theorem 2.1 (Uncertainty bounds)**.** Let X be a discrete finite random variable. Let $L = \operatorname{Card}(X)$.

$$0 \leq H(X) \leq \log(L)$$

Proof.

- For the left equality:

$$H(X) = -\sum_{x \in \operatorname{supp} P_X} P_X(x) \log(P_X(x))$$

Now $P_X(x) \in [0;1]$ and log maps $[0;1]$ to $]-\infty;0]$. Thus $\forall x \in \operatorname{supp} P_X(x)$,

$$-\log(P_X(x)) \geq 0$$

Thus each term is non-negative and the sum is positive.

There is a case of equality if X maps to only one element (thus with probability 1): in this case the uncertainty of X is 0. ($\log(1) = 0$).

- For the right inequality:

$$\begin{aligned}
H(X) &= \sum_x P_X(x) \log(\frac{1}{P_X(x)}) \\
\Leftrightarrow H(X) &= \sum_x P_X(x) \log(\frac{L}{LP_X(x)}) \\
\Leftrightarrow H(X) &= \sum_x P_X(x) \log(L) + \sum_x P_X(x) \log(\frac{1}{LP_X(x)}) \\
\Leftrightarrow H(x) &\leq \log(L) + \sum_x P_X(x)(\frac{1}{LP_X(x)} - 1) \log(e) \\
\Leftrightarrow H(x) &\leq \log(L) + \log(e)(\sum_x \frac{1}{L} - \sum_x P_X(x)) \\
\Leftrightarrow H(x) &\leq \log(L) + \log(e)(1-1) = \log(L)
\end{aligned}$$

□

Now the formula for entropy naturally extends to joint probability distribution since in discrete probability a vector of random variables is just a random variable. More formally: let $(X_i)_{i=1}^n$ be discrete random variable on Ω. Then $X : \omega \in \Omega \mapsto (X_1(\omega), \ldots, X_n(\omega))$ is a random variable (for which in particular uncertainty is defined). Now to simplify notation we are simply going to write:

$$H(X_1, \ldots, X_n) = -\sum_{(x_1, \ldots, x_n)} P_{X_1 \ldots X_n}(x_1, \ldots, x_n) \log(P_{X_1 \ldots X_n}(x_1, \ldots, x_n))$$

Where the sum is indexed by $(X_1(\Omega), \ldots, X_n(\Omega))$

3 Coding the information

Last section, we showed Shannon's measure of information and proved that it held some desired properties. We are now going to show that it is furthermore useful in the context of coding information.

In coding information we mean finding a way to describe efficiently information, i.e. to compress the information to something that can still be traced back to the source output.

3.1 Coding a random variable

We consider a finite discrete random variables U and note $\operatorname{Im} U = \{u_1, \dots, u_n\}$ which is called the *source alphabet*.

Let's first consider the mapping from $W\colon \operatorname{Im} U \to \mathbb{N}_*$ which gives the length of the code of any source alphabet element. Let $M = \max(\operatorname{Im} U)$ be the length of the biggest code length given by the mapping W.

We then consider the *output alphabet* $\mathcal{D}$ which is most of the time considered to be the D-ary alphabet $\{0, \dots, D-1\}$.

Intuitively, we want to assign a different code for each of the elements of the source alphabet. The length of the codes will take their values in $\operatorname{Im} W$ as W describes the length of each code. Also, we consider that the code is a finite sequence of element in the output alphabet, and we can thus define mappings for each of those. More formally we define

$$\forall i \in \mathbb{N}_M, X_i\colon \operatorname{Im} U \to \mathcal{D}$$

which maps any element of the source alphabet to an element of the output alphabet. We will se next why there are M of these mappings. let

$$Z\colon \begin{array}{c} \operatorname{Im} U \to \{(X_i)_{i=1}^k \mid k \in \operatorname{Im} W\} \\ u \mapsto (X_i)_{i=1}^{W(u)} \end{array}$$

which gives the code of any source alphabet element. We now understand that X_i gives the i-th element of the code, and there are M of these mappings because it is the maximum length that any element of the source alphabet can have.

Therefore, if only one element of the source alphabet has a code with maximum length attributed to it, we can consider that X_M should only be

defined for that element and no other in the alphabet. I have not made that clear in the definition of X_i as I felt the notations were already a bit much.

Now, a very interesting thing is that as U is a random variable, the only thing we have done here is to define more random variables: $X_i \circ U$ and $Z \circ U$ each are random variables. Or we can also consider X_i and Z to be random variables on the source alphabet, considering that the element u_i is a message with probability $P_U(u_i)$. We can thus define the average code-length by the expected value of W:

$$\mathrm{E}[W] = \sum_{i=1}^{n} W(u_i) P_U(u_i)$$

Now let $S_{\mathcal{D}} = \{(d_i)_{i=1}^m \,|\, m \in \mathbb{N}, (d_i)_{i=1}^m \in \mathcal{D}^n\}$ be the set of $\mathcal{D}$-*sequences*. We will say that the sequence $z = (d_i)_{i=1}^k$ is a *prefix* of the sequence $z' = (d'_i)_{i=1}^{k'}$ if $k' \geq k$ and $(d_i)_{i=1}^k = (d'_i)_{i=1}^k$. For example a sequence is a prefix of itself. This definition can lead to an interesting property on sets of $\mathcal{D}$-sequences $\{z_1, \ldots, z_k\}$: the set is said to be *prefix-free* if $\forall i \in \mathbb{N}_k, \forall j \in \mathbb{N}_k \backslash \{i\}$, z_i is not a prefix of z_j.

We introduced this notion because we would like $\operatorname{Im} Z$ to be prefix free. Let's see why: let $z \in \operatorname{Im} Z$. We would like to retrieve $u \in \operatorname{Im} U$ such that $Z(u) = z$. Now Z is injective thus there is a unique u that holds this property, so it would seem like there no problem as there is a bijective mapping from $\operatorname{Im} U$ to $\operatorname{Im} Z$. The problem arises when we consider a sequence of codewords: we will use Z to encode *multiple* elements of $\operatorname{Im} U$ and concatenate the result into a message. Now in concatenating we lose the boundaries of our coded information: let's considering $u, u' \in \operatorname{Im} U$ and the message $(Z(u)\ Z(u'))$. In the decoding situation, we have no way of knowing where $Z(u)$ ends and where $Z(u')$ begins.

For example, let $\mathcal{D}$ be {0;1}. And let $\operatorname{Im} U = \{a, b, c\}$ the first three letters of the alphabet. We defined Z such that

$$Z(a) = 1 \qquad Z(b) = 0 \qquad Z(c) = 10$$

The message (a, b) is coded as $(1, 0)$ i.e. 10: it is also how (c) is coded. Thus there is no way to be certain that the message is (c) or (a, b) in the decoding situation, this is what we mean in losing boundaries. That would not happen in a prefix-free mapping, the reason as to why is going to be approached (without formal proof) in the next paragraph.

We define a new mapping to code a sequence of elements of $\operatorname{Im} U$ i.e. a message (of length $k \in \mathbb{N}$): let

$$Z_* : \begin{array}{c} \operatorname{Im}(U)^k \to S_{\mathcal{D}}^k \\ (v_1, \ldots, v_k) \mapsto (Z(v_1), \ldots, Z(v_k)) \end{array}$$

Now $S_{\mathcal{D}}^n$ can still be considered as an element of $S_{\mathcal{D}}$: let

$$((z_{1i})_{i=1}^{k_1}, \ldots, (z_{ni})_{i=1}^{k_n}) \in S_{\mathcal{D}}^k$$

we can observe that

$$(z_{11}, \cdots, z_{1k_1}, \cdots\cdots, z_{n1}, \cdots, z_{nk_n}) \in S_{\mathcal{D}}$$

Let $A \in \operatorname{Im} Z_*$. We will consider an algorithm to decode the information

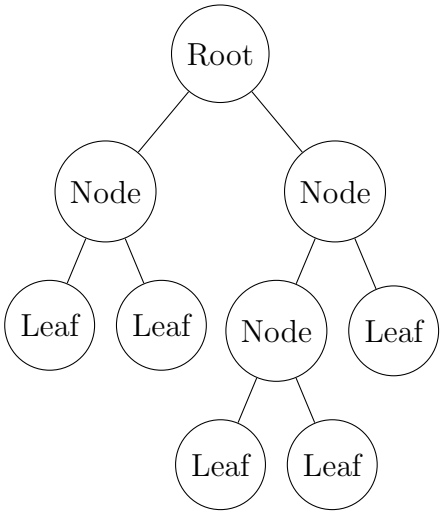
Input: $A \in \operatorname{Im} Z_*$
$originalMessage \longleftarrow \{\}$;
$currentSequence \longleftarrow \{\}$;
while $A \neq \varnothing$ **do**
  Add the first element of A to $currentSequence$;
  Remove first element of A;
  if $currentSequence \in \operatorname{Im} Z_*$ **then**
    let $u \in \operatorname{Im} U$ such that $z(u) = currentSequence$;
    add u to $originalMessage$;
  end
end

As the mapping is prefix-free, you will only find $u \in \operatorname{Im} U$ such that $z(u) = currentSequence$ if u was the original message. If this wasn't the case, then we would a u' (original message) and a u (message actually decoded by the algorithm) such that $Z(u)$ is a prefix of $Z(u')$ by construction, which is absurd.

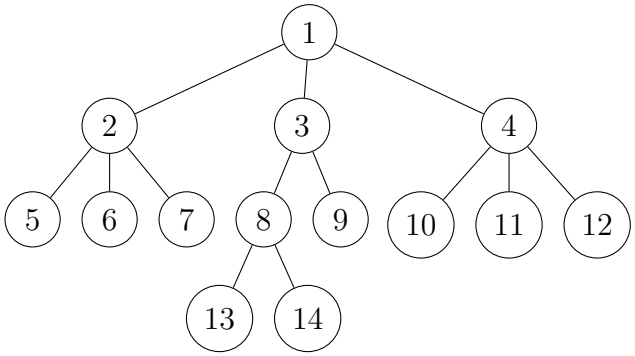
Now that all this nonsense is done, we bring trees to the picture for gaining insights on D-ary codes.

3.2 D-ary Code

We will see that we can represent each output of a mapping on a node of a D-ary tree, which is of the form



Except each node can have at maximum D childs. The above example is a binary-tree. Below we draw a 3-ary tree.



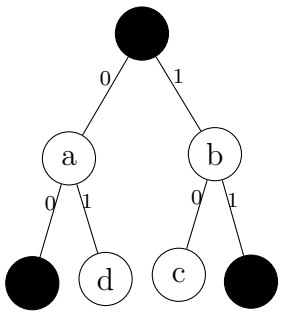
(which has only two childs for some nodes). Now we will try to make the link from an injective mapping

$$Z\colon \operatorname{Im} U \to S_{\mathcal{D}}$$

to a D-ary tree more obvious. Let $\operatorname{Im} U = \{a, b, c, d\}$ and $\mathcal{D} = \{0, 1\}$. Let

$$Z(a) = 0 \qquad Z(b) = 1 \qquad Z(c) = 10 \qquad Z(d) = 01$$

Then the corresponding binary-tree is



Thanks to this representation, we can also see clearly if a code is prefix of another. For that matter, b is a parent node of c and is thus a prefix a c. This is explained by the fact that b is where it is because its code is 1, therefore it is placed on the node of such path (path of edge labeled 1). And c is of code 10, therefore is took the path (1,0) for the edges and is thus right below b.

Now we may want to introduce some more vocabulary before the next theorem. The *level* of a node is the number of edges it would take from the root to get to it. A tree is *full* if there is there are D^N leaves at each level N of the tree. Intuitively, it is full when there is the maximum number of nodes possible. We also introduce the notion of "pruning" a tree at a certain node, which means removing all child subtrees. For example, pruning the tree above at the root node leaves us with juste the root of the tree. Pruning the tree at node b remove the node c and the filled black node on its right (but not the node b). Thus pruning a D-ary tree with depth N at level $w < N$ means removing D^{N-w} nodes.

Theorem 3.1 (Kraft's inequality)**.** There exists a D-ary prefix free code (i.e. a prefix-free injective mapping Z to $S_{\mathcal{D}}$) with code-lengths $\{w_1, \cdots, w_k\} = \operatorname{Im} W$ if and only if

$$\sum_{i=1}^{n} D^{-w_i} \leq 1$$

Proof. Let's suppose we have a prefix-free D-ary code of code-lengths $(w_i)_{i=1}^n$. Let's construct a D-ary tree of such a code (starting with a full tree of depth $N = \max_i w_i$): considering that the code for u_i is $(d_1, \ldots, d_{w_i})$, we place the u_i at path $(d_1, \cdots, d_{w_i})$. Then, we prune the tree at each node u_i. As the code is prefix-free, if in pruning the node u_k (of path $(d'_1, \cdots, d'_{w_k})$) we also deleted a node u_l (of path $(d''_1, \cdots, d''_{w_l})$), this would mean that the path

$(d''_{w_k}, \cdots, d''_{w_l})$ would link the node u_k to the node u_l and thus that

$$(d'_1, \cdots, d'_{w_k}) = (d''_1, \cdots, d''_{w_k})$$

i.e. Z is not prefix free, which is absurd. So considering that at the beginning we had a full binary tree (D^N nodes at each level) and that we were able to prune

$$\sum_{i=1}^{n} D^{N-w_i}$$

and we are still left with a D-ary tree means that the number of nodes we pruned was less than the total number of nodes:

$$\sum_{i=1}^{n} D^{N-w_i} \leq D^N \Leftrightarrow \sum_{i=1}^{n} D^{-w_i} \leq 1$$

Conversely, let's say we have $(w_1, \cdots, w_n)$ such that

$$\sum_{i=1}^{n} D^{-w_i} \leq 1$$

We will construct a prefix-free injective mapping Z where $\operatorname{Im} Z = \{z_1, \cdots, z_n\}$ is such that $W(z_i) = w_i, \forall i \in \mathbb{N}_n$. Now, we could easily rearrange the indexes such that $w_1 \leq \cdots \leq w_n$, and to simplify the notations we will consider it already is that way. Let's consider the following algorithm

Input: $(w_1, \cdots, w_n)$ ordered
Let T be a full D-ary tree of depth w_n;
for $i \leftarrow 1$ **to** n **do**
    // The smallest element in $(w_i, \cdots, w_n)$ is w_i;
    Pick a node at depth w_i;
    Let z_i be the path to that node;
    Prune the tree at that node;
end

Now a crucial step in this algorithm is to be able to pick a node at depth w_i, because there could be none. We are thus going to prove that there is at

least one left at step $i \leq n$. We have at this far pruned nodes at depth w_1 to w_{i-1} i.e. we have pruned

$$\sum_{j=1}^{i-1} D^{w_i - w_j}$$

nodes. Now the number of nodes at depth w_i is D^{w_i} and we have by hypothesis

$$\sum_{k=1}^{n} D^{-w_k} \leq 1 \Leftrightarrow \sum_{k=1}^{n} D^{w_i - w_j} \leq D^{w_i}$$

since each term of the sum is positive we have

$$\sum_{i=1}^{n} D^{w_i - w_j} \geq \sum_{i=1}^{i} D^{w_i - w_j}$$

Thus our hypothesis on $(w_1, \cdots, w_n)$ implies that at each step i of the algorithm, we have not pruned enough nodes to delete every node at depth w_i. Thus the algorithm is correct and finishes which means we have successfully constructed a prefix-free code. □

Now, what we want to achieve is having a minimal average length for our coding map, i.e. minimizing E[W]. The reason as to why should be evident: minimizing means having a coding map that is still prefix-free i.e. decodable, and yet we would need less ressources for sending the information since the length would in average be smaller.

3.3 Lower bound on the average code length

Let's do a quick recap on what exactly we want to do. We have a random variable U, a code-length map W and a prefix-free coding map Z with codomain $S_{\mathcal{D}}$ where $\mathcal{D}$ is the D-ary alphabet. We want to find how much we can reduce the average code-length.

Theorem 3.2.

$$H_D(U) \leq \mathrm{E}[W]$$

Proof. Let's suppose $P_U(\cdot)$ is always non-zero. We have

$$H_D(U) = \sum_{u \in \operatorname{Im} U} P_U(u) \log_D(\frac{1}{P_U(u)})$$

And

$$\mathrm{E}[W] = \sum_{u \in \operatorname{Im} U} P_U(u) W(u)$$

The inequality can be written

$$\begin{aligned}
\sum_u P_U(u) W(u) &\geq \sum_u P_U(u) \log_D(1/P_U(u)) \\
\Leftrightarrow \sum_u P_U(u) \log_D(D^{W(u)}) &\geq \sum_u P_U(u) \log_D(1/P_U(u)) \\
&\Leftrightarrow 0 \geq \sum_u P_U(u)(\log_D(\frac{1}{P_U(u)}) - \log_D(D^{W(u)})) \\
&\Leftrightarrow 0 \geq \sum_u P_U(u) \log_D(\frac{1}{D^{W(u)} P_U(u)}) \\
&\Leftarrow 0 \geq \sum_u P_U(u)(\frac{1}{D^{W(u)} P_U(u)} - 1) \\
&\Leftarrow 0 \geq \sum_u D^{W(u)} - \sum_u P_U(u) \\
&\Leftarrow 1 \geq \sum_u D^{W(u)}
\end{aligned}$$

Kraft's inequlity implies this inequality, which is therefore true. □

Intuitively (and very informally), we can consider the $\log_D$ of a number to be the length of that number in base D. Thus the uncertainty of U ($H_D(U)$) could be considered the average length of the information in $\operatorname{Im} U$ (considering U to take values in $\mathbb{R}$). Considering this, no mapping could describe U in a lesser amount of information (on average) that the information U has.

3.4 Shannon-Fano Codes

Let's go back to the lower bound of the average code-length:

$$\sum_u P_U(u) W(u) \geq \sum_u P_U(u) \log_D(1/P_U(u))$$

Now it would mean that the equality case happens when

$$\forall u \in \operatorname{Im} U,\ W(u) = \log_D(1/P_U(u))$$

Now, $W(u) \in \mathbb{N}$ and $\log_D(1/P_U(u))$ is not always in $\mathbb{N}$, thus we know the equality case can only happen so often. What we might be able to achieve though is

$$W(u) = \lceil \log_D(1/P_U(u)) \rceil$$

And as we know, we can show (and easily construct) a code with such code-length exists by proving that Kraft's inequality is true for this set of code-length.

Proof. $D > 1$ thus $D^{-\mathrm{Id}}$ is strictly decreasing on $[1; +\infty[$. In particular, as $\lceil \log_D(P_U(u)) \rceil \geq \log_D(P_U(u))$

$$\begin{aligned}
\sum_u D^{-\lceil \log_D(1/P_U(u)) \rceil} &\leq \sum_u D^{-\log_D(1/P_U(u))} \\
&\leq \sum_u D^{\log_D(P_U(u))} \\
&\leq \sum_u P_U(u) \\
&\leq 1
\end{aligned}$$

□

Now since

$$\log_D(1/P_U(u)) \leq \lceil \log_D(1/P_U(u)) \rceil \leq \log_D(1/P_U(u)) + 1$$

We what that with such a code

$$H_D(U) \leq \mathrm{E}[W] \leq H_D(U) + 1$$

Which is unfortunately not always optimal.

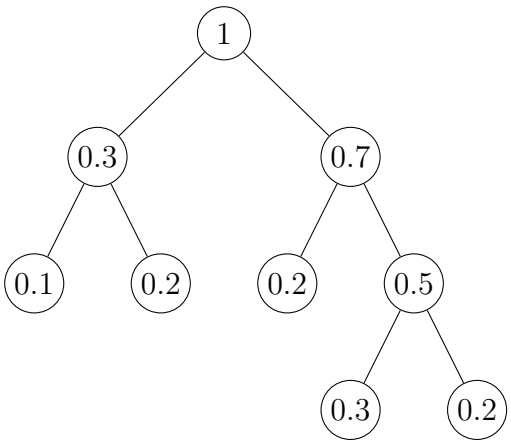
3.5 Optimal Codes: Huffman Code

We are going to deviate from our Kraft's inequality proof algorithm for constructing codes and show that we can construct a code which average code-length is always minimal. We are going to do so using trees with probabilities, which is a tree with probabilities assigned to each vertex such that

- The root has probability 1

- The probability of a parent node is the sum of the probability of its child nodes.

An example of such a tree is given by



Proposition 3.1 (Path length lemma)**.** Let's consider a tree with probabilities. Let $(p_i)_{i=1}^r$ be the probabilities of the leaves, let $(L_i)_{i=1}^r$ be their levels, and let $(q_i)_{i=1}^s$ be the probabilites of the nodes (the vertices that are not leaves). Then

$$\sum_{i=1}^{r} p_i L_i = \sum_{i=1}^{s} q_i$$

In other words, in a tree with probabilities the average level of the leaves is the sum of the probabilities of the nodes.

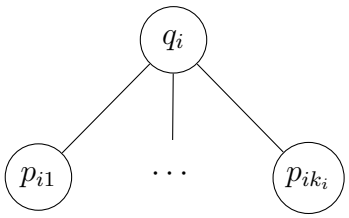
Proof. We prove it by induction on the depth.

- A tree with probabilities of depth 1 has only leaves at level 1 (and at least 1 at that level) thus the average level is 1, which is indeed the probability of the root (axiom 1)

- Let's say the property is true for trees of depth $K-1 \geq 1$ and we want to prove that it's true for trees of depth K. Let T be a K-deep tree. Let's consider the parent nodes of the leaves of T. We note these $(q_i)_{i=1}^t$ and thus the nodes with probabilities $(q_i)_{i=t+1}^s$ are the nodes of the tree that are not a parent of a leaf. Now let's group the leaves by their parent and thus rewrite the probabilites as

$$(p_i)_{i=1}^r = (p_{ij})_{i \in [\![1;s]\!], j \in [\![1;k_i]\!]}$$

where $(p_{ij})_{j\in[\![1;k_i]\!]}$ is the group that has parent node with probability q_i.



We prune the tree at these parent nodes and end up with a $(K-1)$-deep tree. Now, let $(L_i)_{i=1}^t$ be the level of the *parent* nodes of the leaves (i.e. the $(q_i)_{i=1}^s$). The property thus applies and we have

$$\sum_{i=1}^{t} q_i L_i = \sum_{i=t+1}^{s} q_i$$
$$\Leftrightarrow \sum_{i=1}^{t} q_i (L_i + 1) = \sum_{i=1}^{s} q_i$$

Now, because of axiom 2,

$$\forall i \in [\![1;t]\!], q_i = \sum_{j=1}^{k_i} p_{ij}$$

Thus the equality states

$$\sum_{i=1}^{t}\sum_{j=1}^{k_i} p_{ij}(L_i + 1) = \sum_{i=1}^{s} q_i$$

L_i is the level of the parent note of $p_{ij}, \forall j \in [\![1;k_i]\!]$, thus $L_i + 1$ is the level of the leaf p_{ij} and we have our equality for the K-deep tree.

□

We are now able to show that there exists a coding map Z_H (for Huffman Code) with code-length map W_H, such that for all coding map Z and code-length map W,

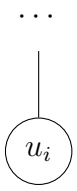
$$\mathrm{E}[W_H] \leq \mathrm{E}[W]$$

We say that the Huffman Code is *optimal* in this sense.

We are going to show the construction of such a code in the case of a binary code (2-ary code). We code a finite random variable U for which $P_U(\cdot) > 0$. Let $\operatorname{Im} U = (u_i)_{i=1}^K$. Let's consider some properties of an optimal code.

Proposition 3.2. In the binary tree of an optimal code, every vertex has a sibling but the root.

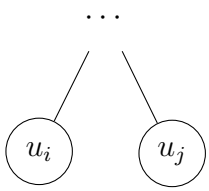
Proof. Let's consider a vertex (u_i) with no sibling. If u_i is a node then we just prune it at that node without loss of code by the prefix-free property.



Then in particular the nodes in the path from the root to the vertex u_i don't have a code assigned to them, by the prefix-free property. Thus we could just consider moving the vertex u_i to its parent node, which would reduce the average length of the code according to the path length lemma. Thus the code would not be optimal which is absurd. □

Proposition 3.3. If a code is optimal, we can transform it into an other optimal code such that the codes for the two least probables messages differ only by the last digit.

Proof. Let's consider two siblings leaves on the deepest level of the tree



Let u_k and u_l be the two least probable messages in $\operatorname{Im} U$.

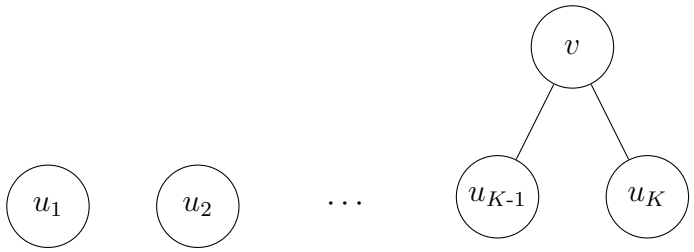
- If $i = k$ we go to next step. Otherwise, as $P_U(u_i) < P_U(u_k)$ and the level of u_k is smaller than the level of u_i, swapping the vertices u_i and u_k does not increase $\mathrm{E}[W]$ by the path length lemma.

- If $j = l$ we have finished the proof. Otherwise, using the same argument we can swap u_j and u_l and still have an optimal code.

□

So now let's actually construct an optimal code: we assume that $(u_1, \cdots, u_K)$ are arranged such that $P_U(u_1) \geq \cdots \geq P_U(u_K)$. We start with

We choose the two least probable messages (in this case u_{K-1} and u_K) and we group them under the same parent node that we note v



Then, we consider the probability assigned with the parent node as $P_U(u_{K-1})+P_U(u_K)$, and we start over with $\{v\} \cup \operatorname{Im} U \setminus \{u_{K-1}, u_K\}$ till we end up with only one node: the root.

4 Conditional entropy

We recall that for random variables $(X_i)_{i=1}^n$, we define the probability distribution over the vector of random variables to be $\forall (x_1, \ldots, x_n) \in X_1(\Omega) \times \cdots \times X_n(\Omega)$,

$$P_{X_1 \ldots X_n}(x_1, \ldots, x_n) = P(\{X_1 = x_1\} \cap \cdots \cap \{X_n = x_n\})$$

We therefore had the following:

$$\sum_{x_i \in X_i(\Omega)} P_{X_1 \ldots X_n}(x_1, \ldots, x_n) = \sum_{x_i \in X_i(\Omega)} P_{X_1 \ldots X_n}(\{X_1 = x_1\} \cap \cdots \cap \{X_n = x_n\})$$

Now a probability distribution is *completly additive* which means that $\forall (E_i)_{i=1}^k$ disjoints, $P(\bigcup_{i=1}^k E_i) = \sum_{i=1}^k P(E_i)$. Thus, as $\forall x \in X_i(\Omega)$,

$$E(x) = \{X_1 = x_1\} \cap \cdots \cap \{X_i = x\} \cap \cdots \cap \{X_n = x_n\}$$

are sets disjoint for one another, we have

$$\begin{aligned}\sum_{x_i \in X_i(\Omega)} P_{X_1 \ldots X_n}(x_1, \ldots, x_n) &= P_{X_1 \ldots X_n}((\bigcap_{\substack{j=1,\\ j\neq i}}^{n} \{X_j = x_j\}) \cap (\bigcup_{x \in X_i(\Omega)} \{X_i = x\}) \\ &= P_{X_1 \ldots X_{i-1} X_{i+1} \ldots X_n}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)\end{aligned}$$

Now this result can be applied to the following: we consider two random variables X, Y and the conditional probability distribution defined for $y \in Y$ such that $P_Y(y) \neq 0$,

$$P_{X|Y}(x|y) = \frac{P_{XY}(x, y)}{P_Y(y)}$$

We thus have

$$\sum_{x \in X(\Omega))} P_{X|Y}(x|y) = \sum_{x \in X(\Omega))} \frac{P_{XY}(x, y)}{P_Y(y)} = \frac{P_Y(y)}{P_Y(y)} = 1$$

and we can consider $P_{X|Y}(\cdot|y)$ to be a probability distribution.

We define for a random variable X, Y, a real-valued function $F\colon X(\Omega) \to \mathbb{R}$ and an event $A \subset X(\Omega) \times Y(\Omega)$ the *conditional expectation* of F given A:

$$\mathrm{E}[F(X)|A] = \sum_{x \in X(\Omega)} F(x) P(X = x|A)$$

So in particular for $A = \{Y = y_c\}$,

$$\mathrm{E}[F(X)|y_c] = \sum_{x \in X(\Omega)} F(x) P_{X|Y}(x|y_c)$$

Or for a function $F\colon X(\Omega) \times Y(\Omega) \to \mathbb{R}$ and an event $A \subset X(\Omega) \times Y(\Omega)$,

$$\mathrm{E}[F(X, Y)|A] = \sum_{x \in X(\Omega)} \sum_{y \in Y(\Omega)} F(x, y) P(\{X = x\} \cap \{Y = y\}|A)$$

Therefore, by choosing $A = \{y = y_0\}$, we have

$$\begin{aligned}
\mathrm{E}[F(X,Y)|Y=y_0] &= \sum_x \sum_y F(x,y) P(\{X=x\} \cap \{Y=y\} | Y=y_0) \\
&= \sum_x F(x,y) P(\{X=x\} \cap \{Y=y_0\} | Y=y_0) \\
&= \sum_x F(x,y) P_{X|Y}(x|y_0)
\end{aligned}$$

The following result is important since it enables for easy computation of the expected value. Starting from the standard expected value,

$$\begin{aligned}
\mathrm{E}[F(X,Y)] &= \sum_y \sum_x P_{XY}(x,y) F(x,y) \\
&= \sum_y \sum_x P_{X|Y}(x|y) P_Y(y) F(x,y) \\
&= \sum_y P_Y(y) \, \mathrm{E}[F(X,Y)|Y=y]
\end{aligned}$$

which is called the *theorem of total expectation*.

We can now introduce the *conditional uncertainty* of X given $Y = y$:

$$H_b(X|Y=y) = -\sum_{x \in \mathrm{supp}(P_{X|Y}(\cdot|y))} P_{X|Y}(x|y) \log_b(P_{X|Y}(x|y))$$

Which can be rewritten

$$H_b(X|Y=y) = \mathrm{E}[-\log_b(P_{X|Y}(X,Y))|Y=y]$$

Now the average (over Y) of the conditional entropy can be computed and is noted $H_b(X|Y)$:

$$\begin{aligned}
H_b(X|Y) &= \sum_y H_b(X|Y=y) P_Y(y) \\
&= \mathrm{E}[-\log_b(P_{X|Y}(X|Y))] \\
&= -\sum_x \sum_y P_{XY}(x,y) \log_b(P_{X|Y}(x|y))
\end{aligned}$$

Theorem 4.1 (Conditioning reduces uncertainty)**.**

$$H_b(X|Y) \leq H_b(X)$$

Proof. First,

$$-\sum_x \sum_y P_{XY}(x,y)\log_b(P_X(x)) = -\sum_x \log_b(P_X(x))\sum_y P_{XY}(x,y)$$
$$= -\sum_x \log_b(P_X(x))P_X(x)$$

Thus there is no difference between taking the average of the function $-\log_b(P_X(x))$ over $X(\Omega)$ or $X(\Omega)\times Y(\Omega)$. In particular, $P_X(x)=0 \Rightarrow P_{XY}(x,y)=0$ and conversely when $P_{XY}(x,y)$ is null, so is $P_X(x)$ or $P_Y(y)$. Either way, they are not included in the set over which we index the sum (supp). We have

$$\begin{aligned}
H_b(X|Y) - H_b(X) &= -\sum_x\sum_y P_{XY}(x,y)\log_b(P_{X|Y}(x|y)) + \sum_x\sum_y P_{XY}(x,y)\log_b(P_X(x)) \\
&= \sum_x\sum_y P_{XY}(x,y)(\log_b(\frac{1}{P_{X|Y}(x|y)}) + \log_b(P_X(x))) \\
&= \sum_x\sum_y P_{XY}(x,y)(\log_b(\frac{P_X(x)}{P_{X|Y}(x|y)})) \\
&= \sum_x\sum_y P_{XY}(x,y)(\log_b(\frac{P_X(x)P_Y(y)}{P_{XY}(x,y)})) \\
\text{by IT-inequality, } &\leq \sum_x\sum_y P_X(x)(\frac{P_X(x)P_Y(y)}{P_{XY}(x,y)} - 1) \\
&= \sum_x\sum_y P_X(x)P_Y(y) - P_{XY}(x,y) \\
&= 0
\end{aligned}$$

i.e. $H_b(X|Y) \leq H_b(X)$. Also, the line 4 of the developement implies that if X and Y are statistically independant, we have $H_b(X|Y) = H_b(X)$. □

This is a very intuitive result because it means that in average (on $Y(\Omega)$), the uncertainty of a random variable given the outcome $Y = y_0$, is less than if we had no data on Y.

Theorem 4.2 (Chain rule for uncertainties)**.** Let $X_1, \ldots, X_n$ be random variables.

$$H_b(X_1 \ldots X_n) = H_b(X_n) + \sum_{i=1}^{n-1} H_b(X_i|X_{i+1}\ldots X_n)$$

Proof. All it takes is to show a certain property of the probability distribution: let's consider X_1 as one random variable and $(X_2, \ldots, X_n)$ as another random variable.

$$P_{X_1(X_2,\ldots,X_n)}(x_1, (x_2, \ldots, x_n)) = P_{X_1|X_2\ldots X_n}(x_1|x_2, \ldots, x_n)P_{X_2\ldots X_n}(x_2, \ldots, x_n)$$

The second part can be decomposed in the same fashion as what we did here, till we are left with

$$P_{X_{n-1}X_n}(x_{n-1}, x_n) = P_{X_{n-1}|X_n}(x_{n-1}|x_n)P_{X_n}(x_n)$$

thus by induction we have the following result

$$P_{X_1\ldots X_n}(x_1, \ldots, x_n) = P_{X_n}(x_n)\prod_{i=1}^{n-1} P_{X_i|X_{i+1}\ldots X_n}(x_i|x_{i+1}, \ldots, x_n)$$

Now taking the $\log_b$:

$$\log_b(P_{X_1\ldots X_n}(x_1, \ldots, x_n)) = \log_b(P_{X_n}(x_n))+\sum_{i=1}^{n-1}\log_b(P_{X_i|X_{i+1}\ldots X_n}(x_i|x_{i+1}, \ldots, x_n))$$

And considering the expression as the function which expected value gives $H_b(X_1 \ldots X_n)$, we have that

$$H_b(X_1 \ldots X_n) = H_b(X_n) + \sum_{i=1}^{n-1} H_b(X_i|X_{i+1} \ldots X_n)$$

Or more simply:

$$H_b(X_1 \ldots X_n) = H_b(X_n) + H_b(X_{n-1}|X_n) + \cdots + H_b(X_1|X_2 \ldots X_n)$$

□

A corollary of that theorem is

Theorem 4.3. Let $X_1, \ldots, X_n$ be random variables.

$$H_b(X_1 \ldots X_n) \geq \sum_{i=1}^{n} H_b(X_i)$$

Proof. We apply the conditioning reduces uncertainty theorem to each element of the development of the last theorem and we obtain the result. □

A last property we could list here is that for two random variables X, Y we have

$$H_b(X|Y) = H_b(X,Y) - H_b(Y)$$

Which is proved by

$$\begin{aligned} H_b(X,Y) - H_b(Y) &= \sum_x \sum_y P_{XY}(x,y) \log_b(\frac{1}{P_{XY}(x,y)}) + \sum_x \sum_y P_{XY}(x,y) \log_b(P_Y(y)) \\ &= \sum_x \sum_y P_{XY}(x,y) \log_b(\frac{P_Y(y)}{P_{XY}(x,y)}) \\ &= \sum_x \sum_y P_{XY}(x,y) \log_b(\frac{1}{P_{X|Y}(x|y)}) \\ &= H_b(X|Y) \end{aligned}$$

Which also implies by the conditioning reduces uncertainty theorem:

$$\begin{aligned} H_b(X,Y) &\geq H_b(Y) \\ H_b(X,Y) &\geq H_b(X) \end{aligned}$$

5 Source coding

5.1 Discrete memory-less source (DMS)

We consider that information sources emit sequences of random variables, where the prefix-free property will show its importance. A *discrete memory-less source* (DMS) is a source that emits an infinite sequence of random variables $(U_i)_{i=1}^{\infty}$ that are statistically independent and identically distributed.

We are going to consider a coding scheme for the DMS where we first *parse* the sequence of random variables: the *parser* is responsible for decomposing the sequence into fixed-length (L). We call such parser an L-block message parser.

$$[U_1 \dots U_L]\,[U_{L+1} \dots U_{2L}] \dots$$

We then encode each block via Huffman's algorithm.

Now we can also consider the source to emit only a finite sequence of random variables at our convenience. For example, taking a sequence of N coin-flips as our source. Each random variable would indeed be identically distributed and the random variables would be statistically independent. Let the i-th flip be represented by S_i. Then the source is very easy to encode as $P_{S_i}(T) = P_{S_i}(H) = 1/2$ and thus $P_{S_1 \dots S_n}(s_1, \dots, s_n) = \frac{1}{2^n}$ (by statistical independance of the random variables, as required by the DMS characterization).

Let V_i be the i-th block formed by the message parser,

$$V_i = [U_{(i-1)L+1} \dots U_{iL}]$$

and let $H(U)$ be the uncertainty of a single message (from the source). We can now answer what is the best average code-length per source message emitted

Theorem 5.1 (Block-to-variable-length coding theorem)**.** There exists a D-ary prefix-free code for a DMS which code-length map W satisfies

$$\frac{\mathrm{E}[W]}{L} \leq H_D(U) + \frac{1}{L}$$

And for every $D - ary$ prefix-free code we have

$$\frac{\mathrm{E}[W]}{L} \geq H_D(U)$$

Proof. Each U_i has the same uncertainty $H(U)$ as they are identically distributed thus

$$H_D(V) = \sum_{i=1}^{L} H_D(U_i) = L \cdot H_D(U)$$

Thus for any D-ary, taking the lower-bound for the code-length of a prefix-free code:

$$\mathrm{E}[W] \geq L \cdot H_D(U) \Leftrightarrow \frac{\mathrm{E}[W]}{L} \geq H_D(U)$$

Now taking the Huffman Code for coding the *block* (or Shannon-Fano's code) which both verify

$$\mathrm{E}[W] \leq H_D(V) + 1 \Leftrightarrow \mathrm{E}[W] \leq L \cdot H_D(U) + 1 \Leftrightarrow \frac{\mathrm{E}[W]}{L} \leq H_D(U)$$

□

Now we might be motivated to consider fixed-length codewords, for example if the codes are stored inside a computer which is very efficient with 64-bits integers, we could consider having binary codewords of length 64. Now our previous work would seem unusable in this situation as we considered code of varying-length, which is actually the property that allowed us to reduce the average code-length. However, they are still relevant if we consider a variable length parser, i.e. *variable-length-to-block* coding of a DMS. We are not going to explore this kind of coding-scheme here.

5.2 Discrete stationary source

We could also consider a Markov Chain to be an information-source, but the information source would not be a DMS as the distribution of the next random variable emitted would depend on the previous random variable i.e. they are not statistically independant. For example, let's consider a Markov-chain with only two possible states like a day being sunny or a day being rainy. Let the probability for a day to have the same weather as the day prior be q. The weather of the first day is uniformly distributed. Then we have for c weather changes

$$P_{S_1,\dots,S_n}(s_1,\dots,s_n) = \frac{1}{2}(1-q)^{n-1-c}q^c$$

i.e. as long as $(s_1,\dots,s_n)$ contains c weather changes, the probability of it happening is as above. Now, we want to compute the uncertainty of S_i and the average entropy of S_i given $S_1 \dots S_{i-1}$. First let's compute $P_{S_i}(s_i)$ for $s_i \in \{R, S\}$. We have

$$P_{S_i}(s_i) = P_{S_i|S_{i-1}}(s_i|R)P_{S_{i-1}}(R) + P_{S_i|S_{i-1}}(s_i|S)P_{S_{i-1}}(S)$$

And by induction, as $P_{S_1}(S) = P_{S_1}(R) = 1/2$,

$$P_{S_i}(s_i) = q \cdot \frac{1}{2} + (1-q) \cdot \frac{1}{2} = \frac{1}{2}$$

Now knowing this, we have that the uncertainty of S_i is 1: it takes 1 bit to code the information. And

$$P_{S_i|S_{i-1}\dots S_1}(s_i|s_{i-1}\dots s_1) = P_{S_i|S_{i-1}}(s_i|s_{i-1})$$

thus the second uncertainty we wanted to compute reduces to

$$\begin{aligned} H_2(S_i|S_{i-1}) &= \sum_{s_i \in \{R,S\}} H_2(S_i|S_{i-1} = s_{i-1}) P_{S_{i-1}}(s_{i-1}) \\ &= \frac{1}{2}(-(1-q)\log_2(1-q) - q\log_2(q)) + \frac{1}{2}(-(1-q)\log_2(1-q) - q\log_2(q)) \\ &= (1-q)\log_2(\frac{1}{1-q}) + q\log_2(\frac{1}{q}) \end{aligned}$$

Now, a *discret stationary source* emits random variables $U_1, U_2, \ldots$ such that $\forall n, L \in \mathbb{N}_*$, the random variables vectors $U_1, \ldots, U_L$ and $(U_{n+1}, \ldots, U_{n+L})$ have the same distribution, i.e. the distribution of the blocks is time invariant. The random variables $U_1, U_2, \ldots$ don't need to be independant, unlike in a DMS, and a DSS can have something called *memory*. It has finite memory $\mu \in \mathbb{N}_*$ if $\forall n > \mu$,

$$P_{U_n|U_1 \ldots U_{n-1}}(u_n|u_1, \ldots, u_{n-1}) = P_{U_n|U_{n-\mu} \ldots U_{n-1}}(u_n|u_{n-\mu}, \ldots, u_{n-1})$$

Intuitively, it means that the DSS only recalls the last μ random variables. In our weather source example, the memory is 1. The conditional uncertainty of the random variables emitted also follows the same property:

$$H_b(U_n|U_{n-1} \ldots U_1) = H_b(U_n|U_{n-1} \ldots U_{n-\mu})$$

Let the *information rate* of a source be

$$H_D^L(U) = \frac{1}{L} H_D(U_1 \ldots U_L)$$

were H^L is not to be taken as an exponent but a notation. This definition might come as a surprise as we could also call information rate the value $H_D(U_L|U_{L-1} \ldots U_1)$, i.e. the uncertainty of each random variable emitted, conditioned by the previous random variables, as we know their outcomes by the memory property. Both are correct as the following theorem states

Theorem 5.2 (Information-theoretic characterization of a DSS)**.**

i. $H_D(U_L|U_1 \ldots U_{L-1})$ is non-increasing with L

ii. $H_D(U_L|U_{L-1} \ldots U_1) \leq H_D^L(U)$, $\forall L \geq 1$

iii. $H_D^L(U)$ is non-increasing with L

iv. $\lim_{L\to\infty} H_D^L(U) = \lim_{L\to\infty} H_D(U_L|U_{L-1}\dots U_1)$

Proof.

i. The probability distribution of the blocks is time-invariant thus

$$H_D(U_L|U_{L-1}\dots U_1) = H_D(U_{L+1}|U_L\dots U_2)$$

now conditioning reduces uncertainty in average thus

$$U_D(U_{L+1}|U_L\dots U_2) \geq H_D(U_{L+1}|U_L\dots U_1)$$

and we have

$$H_D(U_L|U_{L-1}\dots U_1) \geq H_D(U_{L+1}|U_L\dots U_1)$$

ii. We have from the chain rule for uncertainty:

$L\cdot H_D^L(U) = H_D(U_1\dots U_L) = H_D(U_1)+H_D(U_2|U_1)+\dots H_D(U_L|U_{L-1}\dots U_1)$

and by (i) we have that each term of the right side is at least

$$H_D(U_L|U_{L-1}\dots U_1)$$

thus after summing we have

$H_D(U_1)+H_D(U_2|U_1)+\dots H_D(U_L|U_{L-1}\dots U_1) \geq L\cdot H_D(U_L|U_{L-1}\dots U_1)$

And thus by equality

$$H_D^L(U) \geq H_D(U_L|U_{L-1}\dots U_1)$$

iii. We have

$$H_D(U_{L+1}\dots U_1) = H_D(U_{L+1}|U_L\dots U_1) + H_D(U_L\dots U_1)$$

and as

$$H_D(U_{L+1}|U_L\dots U_1) \leq H_D(U_L|U_{L-1}\dots U_1)$$

and

$$H_D(U_L|U_{L-1}\dots U_1) \leq H_D^L(U)$$

we have

$$\begin{aligned} H_D^{L+1}(U) &\leq \frac{1}{L+1}H_D^L(U) + \frac{L}{L+1}\frac{1}{L}H_D(U_L\dots U_1) \\ &= \frac{1}{L+1}H_D^L(U) + \frac{L}{L+1}H_D^L(U) \\ &= H_D^L(U) \end{aligned}$$

iv. $(H_D^L(U))_{L=1}^{\infty}$ and $(H_D(U_L|U_{L-1}\dots U_1))_{L=1}^{\infty}$ are both bounded from below by 0 and non increasing with L, thus their limits exist. Let $H_{D,\infty}(U)$ and $H_{D,\infty}^{cond}(U)$ denote their respective limits. We have $H_{D,\infty}(U) \geq H_{D,\infty}^{cond}(U)$ from (ii). Now by the chain rule we have $\forall n \in \mathbb{N}_*$,

$$H_D(U_1\dots U_{L+n}) = H_D(U_1\dots U_L) + H_D(U_{L+1}|U_1\dots U_L) + \dots + H_D(U_{L+n}|U_1\dots U_{L+n-1})$$

Now

$$H_D(U_{L+1}|U_1\dots U_L)\dots H_D(U_{L+n}|U_1\dots U_{L+n-1}) \leq nH_D(U_{L+1}|U_1\dots U_L)$$

thus combining the two (and dividing by $n+L$) gives

$$H_D^{L+n}(U) \leq \frac{1}{n+L}H_D(U_1\dots U_L) + \frac{n}{n+L}H_D(U_{L+1}|U_L\dots U_1)$$

Now if we take the limit for n, we have on the left side $H_{D,\infty}$. On the right side, the limit of the first term is null and the limit of the second term gives $H_{D,\infty}^{cond}(U)$. We thus have that

$$H_{D,\infty}(U) \leq H_{D,\infty}^{cond}(U)$$

combining that with (ii) gives

$$H_{D,\infty}(U) = H_{D,\infty}^{cond}(U)$$

□

5.3 Lower bound for the average code-length of a DSS

Let's get back to our coding scheme:

$$\text{source} \to L\text{-block parser} \to D\text{-ary encoder}$$

and let's consider as we did before that the i-th block encoded is

$$V_i = [U_{(i-1)L+1}, \dots, U_{iL}]$$

We are going to see a good lower bound for the average code-length W_i of the coding map that encode the blocks V_i. Now, as we have a source with memory

(the following argument still apply if $\mu = 0$), we expect our encoder to take into account the messages emitted before as it could reduce the uncertainty of the block we are currently building a code for and thus change how we construct the code. So given the previous blocks had the messages $(v_i)_{i=1}^{i-1}$, we have

$$H_D(V_i|(V_1 \dots V_{i-1}) = (v_1, \dots, v_{i-1})) = \mathrm{E}[-\log_D(P_{V_i|V_1 \dots V_{i-1}}(\cdot|v_1, \dots, v_{i-1}))]$$

Where the average is indexed over the set of possibilities for the blocks V_i. The same goes for the average code-length

$$\mathrm{E}[W_i|(V_1 \dots V_{i-1}) = (v_1, \dots, v_{i-1})] = \sum_{v_i} W_i(v_i) P_{V_i|V_1 \dots V_{i-1}}(v_i|v_1, \dots, v_{i-1})$$

indexed over the set of possibilities for the blocks V_i. Now as $P_{V_i|V_1 \dots V_{i-1}}(\cdot|v_1, \dots, v_{i-1})$ is a probability distribution the lower bound on the average code-length applies and we have

$$\mathrm{E}[W_i|(V_1 \dots V_{i-1}) = (v_1, \dots, v_{i-1})] \geq H_D(V_i|(V_1 \dots V_{i-1}) = (v_1, \dots, v_{i-1}))$$

Now in our case we are interested in the lower-bound in average (on the set of possibilities for the blocks $V_1 \dots V_{i-1}$), and we have

$$P_{V_i|V_1 \dots V_{i-1}}(v_i|v_1, \dots, v_{i-1}) P_{V_1 \dots V_{i-1}}(v_1, \dots, v_{i-1}) = P_{V_1 \dots V_i}(v_1, \dots, v_i)$$

Thus multiplying each side of the inequality by $P_{V_1 \dots V_{i-1}}(v_1, \dots, v_{i-1})$ gives

$$\sum_{v_i} W_i(v_i) P_{V_1 \dots V_i}(v_1, \dots, v_i) \geq \sum_{v_i} P_{V_1 \dots V_i}(v_1, \dots, v_i) \log_D(\frac{1}{P_{V_1 \dots V_i}(v_1, \dots, v_i)})$$

summing this expression over $(V_1, \dots, V_{i-1})$ gives (by the theorem of total expectations)

$$\mathrm{E}[W_i] = \sum_{v_1, \dots, v_{i-1}} P_{V_1 \dots V_{i-1}}(v_1 \dots v_{i-1}) \, \mathrm{E}[W_i|(V_1 \dots V_{i-1}) = (v_1, \dots, v_{i-1})]$$

and

$$\mathrm{E}[W_i] \geq H_D(V_i|V_{i-1} \dots V_1)$$

Now, we can finally propose a lower bound for the average code-length for the block i:

$$\mathrm{E}[W_i] \geq H_D(V_i|V_{i-1} \dots V_1)$$

$$
\begin{aligned}
&= H_D(U_{(i-1)L+1)} \dots U_{iL} | U_1 \dots U_{(i-1)L}) \\
&= H_D(U_{(i-1)L+1} | U_1 \dots U_{(i-1)L}) + \dots H_D(U_{iL} | U_1 \dots U_{iL-1}) \\
&\geq L \cdot H_D(U_{iL} | U_1 \dots U_{iL-1}) \\
&\geq L \cdot H_{D,\infty}(U)
\end{aligned}
$$

Which gives the lower bound

$$\frac{1}{L} \mathrm{E}[W_i] \geq H_{D,\infty}(U)$$

5.4 Elias-Willems source coding scheme

5.4.1 Coding $\mathbb{N}_*$

We may first solve this problem by just taking the binary representation of the integers, which gives the codemap B such that

$$B(\mathbb{N}_*) = \{1, 10, 11, 100, 101, 110, 111, 1000, \dots\}$$

We will be interested to know the code-length of such a code map. Knowing that a number $n \in \mathbb{N}_*$ has d digits in base b if

$$b^{d-1} \leq n \leq b_d - 1 \Leftrightarrow d - 1 \leq \log_b(n) \leq d \Leftrightarrow d \leq \log_b(n) + 1 \leq d + 1$$

Thus the code length of this representation is $L(n) = \lfloor \log_2(n) \rfloor + 1$. Now this code-map is not prefix-free, which is a property we necessarily want, and that we are going to obtain while maintaining an acceptable code-length.

We first introduce a code map B_1 which takes the code map B and adds $L(n) - 1$ zeros in front of each code-word:

$$B_1(\mathbb{N}_*) = \{1, 010, 011, 00100, 00101, \dots\}$$

Now the code is obviously prefix-free: each code word that had length L now starts with exactly $L(n) - 1$ zeros AND a one right after. This ensures that each code-word of B_1 starts with a a sequence that is unlike any code-word coming before. A downside is that the code-length is

$$L_1(n) = L(n) + L(n) - 1 = 2\lfloor \log_2(n) \rfloor + 1$$

i.e. asymptotically two times longer that our original code-length.

We can again solve this problem by using the code-map we just constructed: let B_2 be a code map which takes the code map B and add in front of the code-word for n: $B_1(L(n))$.

$$B_2(n) = \underbrace{[B_1(L(n))]}_{\text{prefix}} \underbrace{[B(n)]}_{\text{non-prefix}}$$

As B_1 is prefix-free, it means that for each code-word of B_2, we can determine the length of the non-prefix part of the code-word and thus determine where the code-word ends. Now we also don't include the leading zero of $B(n)$ is the non-prefix part of the code map B_2 as it is unnecessary information: we know that if a code-word has length $L(n)$ it has a 1 as leading digit. So for

$$B(n) = (d_1, \dots, d_L) \qquad L = L(n) \qquad B_1(L(n)) = (d'_1, \dots, d'_{L'})$$

We have

$$B_2(n) = d'_1, \dots, d'_{L'}, d_2, \dots, d_L$$

And the code-length is

$$L_2(n) = L(n) - 1 + L_1(L(n)) = \lfloor \log_2(n) \rfloor + 2\lfloor \log_2(\lfloor \log_2(n) \rfloor + 1) \rfloor + 1$$

Which is a better code-length that L_1 considering that $\log_2 \circ \log_2$ is growing slower than $\log_2$.

5.4.2 The coding scheme

We are going to explore a new coding scheme that is going to require the concept of *recency rank*. Let's suppose that we have our usual coding scheme for a DSS:

$$source \to parser \to encoder$$

and let's consider that we have already encoded every possible block at least once ($(\operatorname{Im} U)^L$ is finite so never coding a message would asymptotically be of probability 0). We have the sequence

$$V_1 | \dots | V_i$$

of parsed blocks. To define the recency-rank N_j of the block V_j we consider the biggest $m \geq j$ such that $V_m = V_j$. Then

$$N_j = \operatorname{Card}(\{V_k | k \geq m + 1\})$$

To reason as to why we consider m instead of j is that for the following sequence

$$V_1|V_2|V_3|V_4|V_5 = 01|11|00|01|10$$

we have that $V_1 = 01 = V_4$, and $N_1 = 1$: we don't actually care about the index j of the block, we instead consider the value of the block, so as V_1 and V_4 have the same value their recency rank is the same. Also we can't define N_j to be $i - m$ for the same reasons: we consider the value of the block so taking for example

$$V_1|V_2|V_3 = 01|11|11$$

as V_2 and V_3 are the same blocks, we have that $N_1 = 1$ although V_1 is 2 blocks behind the last parsed block. However, we are going to define $\delta_j = j - m$ as the *indexes to the most recent occurrence of* V_j. We thus obviously have

$$N_j \leq \delta_j$$

We have so far admitted that we are in a situation where every block possible is in the sequence of parsed messages, but it is not a situation we can be in at the beginning of the process. We are going to consider while we are not in this situation that the recency ranks attributed to a block will be arbitrary, until the said block appeared in the sequence of parsed messages and can now be given an actual recency rank.

We are going to assume that the source is *ergodic*, which means that its time statistics is the same as its probability statistics (which is not an accurate nor formal definition of ergodic, but we won't be needing more than this intuition). In other words, if we take a long sequence of emitted messages

$$U_1, \dots U_L$$

And consider the subsequences of length N:

$$U_1, \dots, U_N \qquad U_2, \dots, U_{N+1} \qquad \dots \qquad U_{L-N+1}, \dots, U_L$$

The number of times we are going to a observe a particular message sequence $(x_1, \dots, x_N)$ compared to the number possible sequences:

$$\frac{\text{Card}(\{\text{Subsequences of length } N \text{ that are equal to } (x_1, \dots, x_N)\})}{\text{Card}\{\text{Number of sequences of length } N\}}$$

Is the same as the probability of sequences of length N to be equal to $(x_1, \dots, x_N)$

$$P_{U_1 \dots U_N}(x_1, \dots, x_N)$$

Which seems like a very natural property: what you observe reveals in fact the probability distribution of messages emitted by the source.

We are going to explain now why we want our process to be ergodic: it would mean that the sequences of parsed messages is ergodic (which is trivial), and thus for a block $v \in (\operatorname{Im} U)^L =: V$, we have a probability $P_V(v)$ for the block and by ergodicity, we expect to observe a block of this value every $\frac{1}{P_V(v)}$ which is more formally expressed by

$$\mathrm{E}[\delta_i | V_i = v] = \frac{1}{P_V(v)}$$

Now let's consider the following coding scheme

source (DSS) $\to$ L-block parser $\to$ recency-rank calculator $\to$ encoder

We have that the encoder does not encode the blocks anymore but encodes the recency rank such that for the block V_i:

$$W(V_i) = L_2(N_i) \leq L_2(\delta_i)$$

Now

$$\begin{aligned} L_2(\delta_i) &= \lfloor \log_2(\delta_i) \rfloor + 2\lfloor \log_2(\lfloor \log_2(\delta_i) \rfloor + 1) \rfloor + 1 \\ &\leq \log_2(\delta_i) + 2\log_2(\log_2(\delta_i) + 1) + 1 \end{aligned}$$

We have from Jensen's inequality:

$$\mathrm{E}[W(V_i)|V_i = v] \leq \log_2(\mathrm{E}[\delta_i|V_i = v]) + 2\log_2(\log_2(\mathrm{E}[\delta_i|V_i = v]) + 1) + 1$$

We can finally use ergodicity:

$$\mathrm{E}[W(V_i)|V_i = v] \leq -\log_2(P_V(v)) + 2\log_2(-\log_2(P_V(v)) + 1) + 1$$

Using the theorem of total expectation (i.e. summing over V and multiplying by $P_V(v)$):

$$\mathrm{E}[W(V_i)] \leq H_2(V) + 1 + \sum_v P_V(v) \log_2(\log_2(P_V(v)) + 1)$$

And using Jensen's inequality:

$$\mathrm{E}[W(V_i)] \leq H_2(V) + \log_2(H_2(V) + 1) + 1$$

Now, $H_2(V) = H_2(U_1 \dots U_L) = L \cdot H_2^L(U)$, we thus have the following theorem

Theorem 5.3 (Elias-Willems block-to-variable-length coding theorem for a DSS)**.**

$$\frac{\mathrm{E}[W(V_i)]}{L} \leq H_2^L(U) + \frac{2}{L}\log_2(L \cdot H_2^L(U) + 1) + \frac{1}{L}$$

Now the above theorem states a code-length that is not as good as the code-length for a DSS with a Huffman encoder. So why would be use the above coding scheme instead? The above coding scheme has a very interesting property which is that it is universal: it works well for all random variable *and not only the ones for which we know the underlying probability distribution.*

Intuitively, the Elias-Willems coding scheme learns from the code thanks to its recency-rank calculator while its encoding information. Thus, although it's less efficient that Huffman Code, we *can* use it for encoding information we know nothing about.