Introduction to Information Theory

Last Updated February 16th, 2024

1 Entropy of an Information Source

An information source is a pair $S = (\mathcal{A}, P)$ where $\mathcal{A} = \{x_1, \ldots, x_n\}$ is called the **source alphabet** and P is a **probability distribution** on $\mathcal{A}$, i.e. $P = \{p_1, \ldots, p_n\}$, where p_i denotes the probability of x_i. We also denote p_i as $p(x_i)$.

Example 1.1. When Heather brings home a math test to show her parents, the exam's letter grade follows the following probability distribution: A: 0.60, B: 0.25, C: 0.10, D: 0.05. Thus, the letter grade on Heather's math test represents an information source with

$$\mathcal{A} = \{\text{A,B,C,D}\},$$

and

$$P = 0.60, 0.25, 0.10, 0.05.$$

Our goal is to now quantitatively answer the following question: on average, how much information does an information source actually provide? If we think of a news event, the amount of interest in the event involves two factors: the event's psychological impact on a human receiver, and its likelihood of occurring. In this lecture we focus on the latter factor for measuring information content since i) psychological impact seems quite subjective in its measurement and ii) we wish our definition to apply to situations for which a human is not the receiver of the information, and thus there is no pychological-impact factor to consider. In this case, the following statement seems supported by everyday experience:

"The less likely an event, the more information that is conveyed by the event."

Thus, assuming p is the probability that an event occurs, we seek a function $I(p)$ defined on the interval $(0, 1]$ that has the following properties.

1. $I(p)$ increases as p decreases
2. $I(1) = 0$
3. $\lim_{p \to 0+} I(p) = \infty$

Property 2 states that, if an event is certain to occur, then its occurrence conveys zero information, while Property 3 states that there is no bound on the amount of information that an event may convey so long as its likelihood becomes increasingly small.

Definition 1.2. Given an information source $S = (\mathcal{A}, P)$, where $\mathcal{A} = \{x_1, \ldots, x_n\}$ and $P = \{p_1, \ldots, p_n\}$, the **entropy** of S, written $H(p_1, \ldots, p_n)$, equals the *average* amount of information conveyed by a symbol of $\mathcal{A}$, i.e.

$$H(p_1, \ldots, p_n) = \sum_{i=1}^{n} p_i I(p_i).$$

1.1 Deriving $I(p)$

We now turn our attention to deriving a mathematical formula for $I(p)$. We accomplish this by stating the following three properties that we believe $H(p_1, \ldots, p_n)$ should possess.

1. $H(p_1, \ldots, p_n)$ is continuous over the n-dimensional cube $[0, 1]^n$ subject to the constraint that $x_1 + \cdots + x_n = 1$.

   Rationale: entropy should react in a continuous manner to continuous changes of probabilities.

2. $H(\frac{1}{n}, \ldots, \frac{1}{n}) < H(\frac{1}{n+1}, \ldots, \frac{1}{n+1})$.

   Rationale: since $H(\frac{1}{n}, \ldots, \frac{1}{n}) = I(\frac{1}{n})$, $H(\frac{1}{n+1}, \ldots, \frac{1}{n+1}) = I(\frac{1}{n+1})$ and, by Property 1 of $I(p)$ stated above, we must have $I(\frac{1}{n}) < I(\frac{1}{n+1})$.

3. $H(\frac{1}{n}, \ldots, \frac{1}{n}) = H(\frac{b_1}{n}, \ldots, \frac{b_k}{n}) + \sum_{i=1}^{k} \frac{b_i}{n} H(\frac{1}{b_i}, \ldots, \frac{1}{b_i})$, where $\sum_{i=1}^{k} b_i = n$.

   Rationale: to observe one of $\mathcal{A}$'s symbols, we obtain k empty bags, and divide up the symbols into k disjoint sets $S_1, \ldots, S_k$, where $|S_i| = b_i$. We then place the symbols in S_i into the i th bag. Finally, we randomly select one of the bags, where the i th bag has probability $\frac{b_i}{n}$ of being selected, and randomly (according to a uniform distribution) select one of the symbols from the bag. Thus, we desire that the entropy of the source equals the sum of i) the average amount of information conveyed by the selection of a particular bag, and ii) the information conveyed by selecting a particular symbol from that bag.

Theorem 1.3. The only way to define $I(p)$ and satisfy the above three properties is for

$$I(p) = -\log_b p = \log_b \left(\frac{1}{p}\right),$$

where $b \geq 2$ can be any base. Consequently,

$$H_b(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_b p_i = \sum_{i=1}^{n} p_i \log_b \left(\frac{1}{p}\right).$$

When $b = 2$ the unit of entropy is **bits** and in this case we drop the subscript and write $H(p_1, \ldots, p_n)$.

Example 1.4. Returning to Example 1.1, the average amount of information conveyed by the letter grade on Heather's math test equals

$$H(0.60, 0.25, 0.10, 0.05) = 0.60 \log\left(\frac{1}{0.60}\right) + 0.25 \log\left(\frac{1}{0.25}\right) + 0.10 \log\left(\frac{1}{0.10}\right) +$$

$$0.05 \log\left(\frac{1}{0.05}\right) = 1.49 \text{ bits.}$$

Example 1.5. The 12-balls puzzle. You are given 12 balls, all of which are identical in size and weight, except one, which may be heavier or lighter than the other 11. You must determine which ball is non-standard using only a balance that can support up to six balls on each side. Determine the initial balancing experiment that conveys the most information in the worst case.

2 Encoding an Information Source

Given an information source $S = (\mathcal{A}, P)$, we may think of a symbol $x_i \in \mathcal{A}$ as representing an event and its associated probability p_i as the likelihood of the event occurring. Furthermore, when the event occurs, we may want to communicate its occurrence which involves using a *commmunication channel*. Finally, we prefer to be as efficient as possible in the sense that, if the communication involves sending a string of characters through the channel, then the average string length should be made as small as possible without compromising communication accuracy. In this lecture we assume that the channel is **noiseless** in the sense that whatever string is sent through the channel is the same string that is read by the receiver.

Definition 2.1. A b-ary **code** is a set of words C over some alphabet Ψ, where $|\Psi| = b$. The members of $\mathcal{C}$ are called **codewords**. When $b = 2$, $\mathcal{C}$ is called a **binary code**, when $b = 3$, $\mathcal{C}$ is called a **ternary code**.

Definition 2.2. Given a set Z, Z^+ denotes the set of all sequences of the form $z_1 \cdots z_k$, where $k \geq 1$, and $z_i \in Z$ for all $1 \leq i \leq k$.

Definition 2.3. A b-ary **encoding** of an information source $S = (\mathcal{A}, P)$ is a one-to-one function $f : \mathcal{A} \to \mathcal{C}$, where $\mathcal{C} = \Psi^+$ and $|\Psi| = b$. The code associated with the encoding is

$$C = \{c_1 = f(x_1), \ldots, c_n = f(x_n)\}.$$

Definition 2.4. Given an encoding function $f : \mathcal{A} \to \mathcal{C}$ for an information source, the **extension of** f is the function $f^+ : \mathcal{A}^+ \to \mathcal{C}^+$ for which

$$f^+(x_{i_1} \cdots x_{i_k}) = f(x_{i_1}) \cdots f(x_{i_k}).$$

In other words, f^+ is mapping words over the alphabet $\mathcal{A}$ to words over the alphabet $\mathcal{C}$ (here, each codeword is being viewed as a symbol).

Definition 2.5. An encoding $f : \mathcal{A} \to \mathcal{C}$ is said to be **uniquely decodable** iff f^+ is a one-to-one function. In other words, for any sequence of codewords, there is at most one word in $\mathcal{A}^+$ that maps to that sequence.

Note that, although unique decodability is a property of an encoding, we also associate it with the code itself. For example, if $C = \{c_1, \ldots, cn\}$, then C is **uniquely decodable** iff for every word $w \in C^+$, there is a unique sequence $c_{i_1} \cdots c_{i_k}$ for which $w = c_{i_1} \cdots c_{i_k}$.

Definition 2.6. A code C is said to be a **prefix code** iff no codeword in $\mathcal{C}$ is a prefix of some other codeword in C. In other words, there are no $u, w \in \mathcal{C}$ for which $u \cdot v = w$, for some $v \in \mathcal{C}$.

We leave the following as an exercise.

Proposition 2.7. Every prefix-code is uniquely decodable.

Example 2.8. The code $C = \{0, 10, 110, 111\}$ is a prefix code, however $C' = \{22, 321, 233, 13, 221\}$ is not a prefix code since 22 is a prefix of 221.

Example 2.9. Let $\mathcal{A} = \{a, b, c, d\}$ and consider the binary encoding defined by $f(a) = 0$, $f(b) = 01$, $f(c) = 011$, and $f(d) = 0111$. Then the resulting code $C = \{0, 01, 011, 0111\}$ is not a prefix code since, e.g., 0 is a prefix of every codeword. However, the encoding is still uniquely decodable since, when a 0 is read, counting the subsequent 1's that follow, up to the next 0, identifies the unique codeword.

2.1 Probability encodings

As we'll see, the results of this lecture depend only the relationship between codewords and probabilities, and are independent of the information source. For this reason we may define a **probability (distribution) encoding** as a pair $E = (C, P)$, where $C = \{c_1, \ldots, c_n\}$ is a set of codewords, and $P = \{p_1, \ldots, p_n\}$. Note that an encoding f of an information source $S = (\mathcal{A} = \{x_1, \ldots, x_n\}, P)$ always induces a probability encoding $E = (C, P)$, where $C = \{f(x_1), \ldots, f(x_n)\}$.

Definition 2.10. Given a probability encoding $E = (C, P)$, the **average codword length**, denoted $\text{AveLen}(C, P)$, is given by

$$\text{AveLen}(C, P) = \sum_{i=1}^{n} p_i |c_i|.$$

Example 2.11. Given the probability encoding $E = (C = \{0, 01, 011, 0111\}, P = \{0.5, 0.25, 0.125, 0.125\})$, the average codeword length equals

$$\text{AveLen}(C, P) = \sum_{i=1}^{n} p_i |c_i| = (0.5)(1) + (0.25)(2) + (0.125)(3) + (0.125)(4) = 1.875.$$

2.2 Kraft's Inequality

Theorem 2.12. (Kraft's Inquality) For any b-ary prefix-code C with codeword lengths $l_1, \ldots, l_n$,

$$\sum_{i=1}^{n} b^{-l_i} \leq 1.$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists a b-ary prefix code with these word lengths.

Proof of Kraft's Inequality WLOG, assume that the codeword lengths are in non-decreasing order and Consider a perfect b-ary tree $\mathcal{T}$ with height l_n. The we have the following basic facts.

1. $\mathcal{T}$ has b^{l_n} leaves
2. there is a one-to-one correspondence between b-ary words of length not exceeding l_n and nodes of $\mathcal{T}$
3. there is one-to-one mapping from C into the set of nodes of $\mathcal{T}$
4. every leaf of $\mathcal{T}$ has at most one ancestor in C
5. every codeword $c_i \in C$ is the ancestor of exactly $b^{l_n - l_i}$ leaves of $\mathcal{T}$

From the above facts we see that

$$\sum_{i=1}^{n} b^{l_n - l_i} \leq b^{l_n},$$

and hence (dividing both sides by b^{l_n})

$$\sum_{i=1}^{n} b^{-l_i} \leq 1.$$

Conversely, suppose that lengths $l_1, \ldots, l_n$ satisfy

$$\sum_{i=1}^{n} b^{-l_i} \leq 1.$$

Let $\mathcal{T}$ be a perfect b-ary tree. Define binary code C in the following manner.

Basis step. Let $w_1 = 0^{l_1}$.

Inductive step. Assume that there exists $1 \leq k \leq n-1$ such that codewords $w_1, \ldots, w_k$ have been defined in such a way that $|w_i| = l_i$, for all $1 \leq i \leq k$, and that the codewords represent a prefix code. Then by Kraft's inequality,

$$\sum_{i=1}^{k} b^{-l_i} < 1,$$

which implies

$$\sum_{i=1}^{k} b^{l_n - l_i} < b^{l_n}.$$

Then there exists a leaf L of $\mathcal{T}$ for which no member of $\{w_1, \ldots, w_k\}$ is an ancestor of L. Choose the first such leaf L and set w_{k+1} to the ancestor of L having length l_{k+1}. Continuing in this manner a prefix-code C with the desired word lengths will be attained. □

It's interesting to note that a theorem by McMillan states that Kraft's inequality still holds in case $\mathcal{C}$ is uniquely decodable (but not necessarily a prefix code). One corollary of McMillan's theorem is that, for every uniquely decodable code $\mathcal{C}$, there is a prefix code $\mathcal{C}'$ whose codeword lengths match those of $\mathcal{C}$. Thus, no efficiency is lost (in terms of minimizing average codeword length) by working exclusively with prefix codes.

Example 2.13. Is there a uniquely decodable code over the alphabet $\{0, 1, \ldots, 9\}$ with 9 codewords of length 1, 9 codewords of length 2, 10 codewords of length 3, and 10 codewords of length 4?

Solution.

3 Huffman's Coding Algorithm

For the sake of clarity, in this section we assume the use of binary codewords. With that said, Huffman's algorithm works for codeword alphabets of any size ≥ 2. For sizes other than 2, slight changes must be made to both the algorithm and its proof of correctness. The details are left as an exercise.

The **Huffman Coding Algorithm** is a recursive greedy algorithm that computes an optimal prefix code for a probability distribution $\mathcal{P} = \{p_1, \ldots, p_n\}$, where

$$p_1 \geq \cdots \geq p_{n-1} \geq p_n.$$

In what follows we let $h(p_i)$ denote the codeword that Huffman's algorithm assigns to probability p_i, $i = 1, \ldots, n$.

Base Case If $\mathcal{P} = \{1.0\}$, then $h(1.0) = \lambda$.

Recursive Case Assume $\mathcal{P} = \{p_1, \ldots, p_n\}$ with $n \geq 2$.

Recursively call Huffman on

$$\mathcal{P}' = \{p_1, \ldots, p_{n-2}, p_{n-1} + p_n\}.$$

Return

$$\mathcal{C} = \{h(p_1), \ldots, h(p_{n-2}), h(p_{n-1} + p_n) \cdot 0, h(p_{n-1} + p_n) \cdot 1\},$$

meaning that the code returned by Huffman from the recursive call is expanded by replacing codeword $h(p_{n-1} + p_n)$ with the two sibling codewords $h(p_{n-1} + p_n) \cdot 0$ and $h(p_{n-1} + p_n) \cdot 1$.

Example 3.1. Apply Huffman's algorithm to the probability distribution $\{0.3, 0.23, 0.22, 0.15, 0.10\}$.

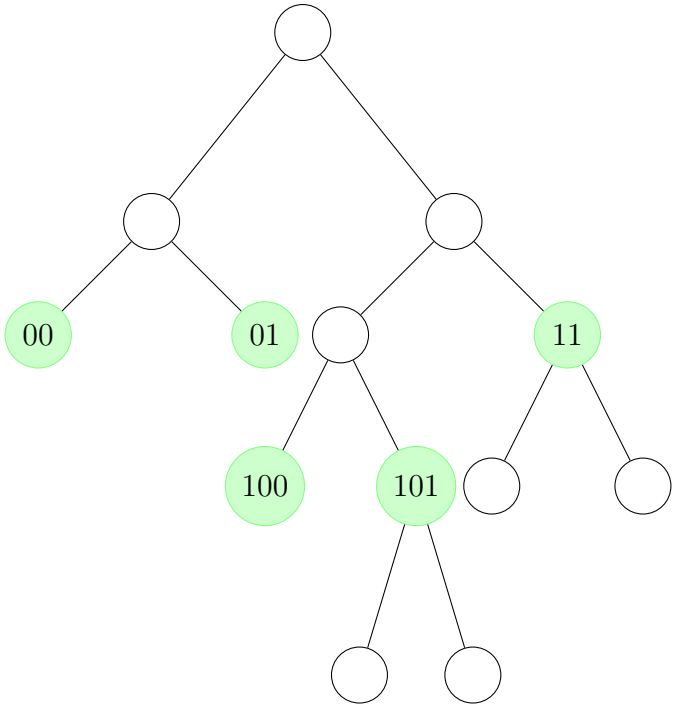

Figure 1: Code tree for prefix code $\mathcal{C} = \{00, 01, 100, 101, 11\}$

It helps to think of the codewords of a binary prefix code as nodes on a binary tree. For example, the codeword 1 represents the right child of the tree root, while 01 represents the right child of the left child of the tree root (or the root's left-right grandchild). Moreover, being a prefix code means that no codeword can be an ancestor of any other codeword. Figure 1 shows a binary code tree for the prefix code $\mathcal{C} = \{00, 01, 100, 101, 11\}$.

Claim: for any probability distribution there is an optimal prefix code for which the two least probable codewords are (tree) siblings.

Proof of Claim: WLOG, assume that the respective codeword probabilities are $p_1 \geq \cdots \geq p_{n-1} \geq p_n$. Suppose w_{n-1} and w_n are the two least probable codewords of an optimal prefix code $\mathcal{C}$. First notice that w_{n-1} and w_n must be the two longest codewords. For suppose codeword w_i has a length that exceeds $\max(|w_{n-1}|, |w_n|)$. If this were true then, since $p_i \geq p_{n-1} \geq p_n$, we may assign p_i codeword w_n, and p_n codeword w_i, resulting in an average codeword length that does not exceed the optimal average length (show this!), and so the new probability encoding must also be optimal.

The next observation is that we must have $|w_{n-1}| = |w_n|$. For suppose $|w_n| > |w_{n-1}|$, i.e., w_n exists at a lower level of the tree than that of w_{n-1}. Then w_n is the only codeword at this level (why?), and hence its parent is not the ancestor of any other codeword. Thus, w_n may be replaced by its parent to obtain a prefix code of lesser average length, a contradiction.

Finally, given that w_{n-1} and w_n reside at the same (bottom) tree level, if w_n has no sibling codeword, then we may replace w_{n-1} with w_n's sibling, to obtain another (optimal) code having the same average length. On the other hand, if, say, codeword w_{n-2} is the sibling of w_n, then we may swap the codewords of p_{n-1} and p_{n-2} to obtain another (optimal) code in which the two least probable codewords are siblings. □

Theorem 3.2. Huffman's algorithm is correct in that it always returns a probability encoding of $P = \{p_1, \ldots, p_n\}$ that has the least average codeword length.

Proof. We use mathematical induction.

Basis step. if $n = 1$, then $\mathcal{P} = \{1.0\}$, and $h(1.0) = \lambda$ is optimal, since the average codeword length equals $0(1) = 0$.

Inductive step Part 1. Assume that, for some $n \geq 2$, Huffman's algorithm always returns an optimal prefix code for probability distributions with $n - 1$ or fewer members. Now consider the input $\mathcal{P} = \{p_1, \ldots, p_{n-1}, p_n\}$. Then, by the inductive assumption, the recursive call to Huffman on input $\{p_1, \ldots, p_{n-1} + p_n\}$ results in returning an optimal prefix code which we'll call $\mathcal{C}_{\text{huff,opt},n-1}$. This code is then expanded to a code for $\mathcal{P}$ by replacing $h(p_{n-1} + p_n)$ with $h(p_{n-1} + p_n) \cdot 0$, and $h(p_{n-1} + p_n) \cdot 1$. Call this new code $\mathcal{C}_{\text{huff},n}$.

Notice that $\mathcal{C}_{\text{huff},n}$ has an average codeword length that is only $p_{n-1} + p_n$ more than the average for $\mathcal{C}_{\text{huff,opt},n-1}$. This is because the expansion of $\mathcal{C}_{\text{huff,opt},n-1}$ replaces $h(p_{n-1} + p_n)$ with two words, each of which is only one bit longer than $h(p_{n-1} + p_n)$, thus adding an extra

$$1(p_{n-1}) + 1(p_n) = p_{n-1} + p_n$$

in average codeword length.

To summarize, we have

$$L(\mathcal{C}_{\text{huff},n}) = L(\mathcal{C}_{\text{huff,opt},n-1}) + p_{n-1} + p_n.$$

Inductive Step Part 2. Now consider an optimal prefix code $\mathcal{C}_{\text{opt},n}$ for $\mathcal{P}$ in which the two least-probable codewords are siblings. Then letting

$$\mathcal{P}' = \{p_1, \ldots, p_{n-1} + p_n\},$$

we may use $\mathcal{C}_{\text{opt},n}$ to create the code $\mathcal{C}_{n-1}$ with the only change being the replacement of codewords w_{n-1} and w_n with their longest common prefix y. And since

$$|y| = |w_{n-1}| - 1 = |w_n| - 1,$$

This yields

$$L(\mathcal{C}_{\text{opt},n}) = L(\mathcal{C}_{n-1}) + p_{n-1} + p_n.$$

Thus, we have established the two following facts.

1. The optimal prefix code $\mathcal{C}_{\text{huff,opt},n-1}$ yields the code $\mathcal{C}_{\text{huff},n}$ whose average length is $p_{n-1} + p_n$ more than that of $\mathcal{C}_{\text{huff,opt},n-1}$.
2. The optimal prefix code $\mathcal{C}_{\text{opt},n}$ yields a prefix code $\mathcal{C}_{n-1}$ whose length is $p_{n-1} + p_n$ less than that of $\mathcal{C}_{\text{opt},n}$.

The above two facts imply that

$$L(\mathcal{C}_{\text{huff},n}) - L(\mathcal{C}_{\text{huff,opt},n-1}) = L(\mathcal{C}_{\text{opt},n}) - L(\mathcal{C}_{n-1}),$$

which in turn implies that

$$L(\mathcal{C}_{\text{huff},n}) = L(\mathcal{C}_{\text{opt},n}),$$

meaning that $\mathcal{C}_{\text{huff},n}$ is optimal (since $\mathcal{C}_{\text{opt},n}$ is optimal). To see why this must be true, suppose instead we have

$$L(\mathcal{C}_{\text{huff},n}) > L(\mathcal{C}_{\text{opt},n}).$$

Then the above equation would force

$$L(\mathcal{C}_{\text{huff,opt},n-1}) > L(\mathcal{C}_{n-1}),$$

which is a contradiction, since $\mathcal{C}_{\text{huff,opt},n-1}$ is assumed optimal by the inductive assumption. Therefore, $\mathcal{C}_{\text{huff},n}$ is optimal and, since this is the code returned by Huffman for an input of size n, we see that Huffman's algorithm is correct. □

Example 3.3. The following table shows each subproblem that is solved by Huffman's algorithm for the problem instance provide in Example 3.1, its optimal code, the code's average length, and how the difference in average length between a parent and child code is equal to the sum of the two least probabilities of the parent code.

n	**Prob**	**Code**	$L(C_i)$	$L(C_i) - L(C_{i-1})$
1	$\{1.0\}$	$\{\lambda\}$	0	
2	$\{0.55, 0.45\}$	$\{0, 1\}$	1	$1 - 0 = 0.55 + 0.45$
3	$\{0.45, 0.30, 0.25\}$	$\{1, 00, 01\}$	1.55	$1.55 - 1 = 0.30 + 0.25$
4	$\{0.30, 0.25, 0.23, 0.22\}$	$\{00, 01, 10, 11\}$	2	$2 - 1.55 = 0.23 + 0.22$
5	$\{0.30, 0.23, 0.22, 0.15, 0.10\}$	$\{00, 10, 11, 010, 011\}$	2.25	$2.25 - 2 = 0.15 + 0.1$

4 The Noiseless Coding Theorem

We begin with the Log-Concavity Lemma which gets its name from the fact that $\ln x$ is an increasing concave function, and thus satisfies the inequality

$$\ln x \leq x - 1 \tag{1}$$

for all x, with equality iff $x = 1$. This is true since $y = x - 1$ is tangent to $\ln x$ at $x = 1$ (draw the graph for each function).

Lemma 4.1. (Log-Concavity Lemma) Let $P = \{p_1, \ldots, p_n\}$ be a probability distribution and let $Q = \{q_1, \ldots, q_n\}$ have the properties i) $0 \leq q_i \leq 1$ and ii) $\sum_{i=1}^{n} q_i \leq 1$. Then

$$\sum_{i=1}^{n} p_i \log \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \log \frac{1}{q_i},$$

with equality iff $p_i = q_i$ for all $i = 1, \ldots, n$.

Proof. Since the proof is independent of the log base, we will work with natural logarithms.

Claim: for all $i = 1, \ldots, n$

$$p_i \ln \frac{1}{p_i} \leq p_i \ln \frac{1}{q_i} + q_i - p_i.$$

Indeed, if $p_i = 0$, then the above inequality reduces to $q_i \geq 0$ which is true, and if $p_i \neq 0$ but $q_i = 0$, then the left side is finite while the right side equals ∞. Finally, if both $p_i > 0$ and $q_i > 0$, then

$$p_i \ln \frac{1}{p_i} \leq p_i \ln \frac{1}{q_i} + q_i - p_i$$

is true iff

$$p_i \ln \frac{q_i}{p_i} \leq q_i - p_i$$

which in turn (upon dividing both sides by p_i) is true iff

$$\ln \left(\frac{q_i}{p_i} \right) \leq \left(\frac{q_i}{p_i} \right) - 1,$$

which is true by 1 above, and with equality iff $q_i = p_i$.

To finish the proof, if we sum the claimed statement over $i = 1, \ldots, n$, we get

$$\sum_{i=1}^{n} p_i \ln \frac{1}{p_i} \leq \sum_{i=1}^{n} p_i \ln \frac{1}{q_i} + \sum_{i=1}^{n} q_i + \sum_{i=1}^{n} p_i \leq \sum_{i=1}^{n} p_i \ln \frac{1}{q_i}$$

with equality holding iff $q_i = p_i$ for all $i = 1, \ldots, n$. □

Theorem 4.2. Let $E = (C, P)$ be a probability encoding, where $C = \{c_1, \ldots, c_n\}$ and $P = \{p_1, \ldots, p_n\}$. Then

$$H_b(p_1, \ldots, p_n) \leq \text{AveLen}(C, P)$$

with equality iff, for all $i = 1, \ldots, n$,

$$|c_i| = \log_b \left(\frac{1}{p_i}\right).$$

Proof. Since C is an instantaneous code, by Kraft's inequality, we have

$$\sum_{i=1}^{n} \frac{1}{b^{l_i}} \leq 1.$$

Then by the Log-Concavity Lemma, and letting $q_i = \frac{1}{b^{l_i}}$, we have

$$H_b(p_1, \ldots, p_n) = \sum_{i=1}^{n} p_i \log_b \left(\frac{1}{p_i}\right) \leq \sum_{i=1}^{n} p_i \log_b \left(\frac{1}{q_i}\right) =$$

$$\sum_{i=1}^{n} p_i \log_b b^{l_i} = \sum_{i=1}^{n} p_i l_i = \text{AveLen}(c_1, \ldots, c_n).$$

Finally, by the Log-Concavity Lemma, equality holds iff $p_i = q_i$, for $i = 1, \ldots, n$, which is true iff

$$l_i = \log_b \left(\frac{1}{p_i}\right)$$

for $i = 1, \ldots, n$. □

Theorem 4.3. (Noiseless Coding Theorem) For any probability distribution $P = \{p_1, \ldots, p_n\}$, let MinAveLen(P) denote the minimum average codeword length attained over all possible probability encodings of P. Then

$$H_b(p_1, \ldots, p_n) \leq \text{MinAveLen}(P) < H_b(p_1, \ldots, p_n) + 1.$$

Proof. The first inequality is the result of Theorem 4.2. To show the upper bound, consider the following lengths

$$l_i = \lceil \log_b \frac{1}{p_i} \rceil.$$

These lengths satisfy the Kraft inequality since

$$\sum_{i=1}^{n} b^{-\lceil \log_b \frac{1}{p_i} \rceil} \leq \sum_{i=1}^{n} b^{-\log_b \frac{1}{p_i}} = \sum_{i=1}^{n} p_i = 1.$$

Thus, by Theorem 2.12, there is a prefix code $\mathcal{C}$ whose respective codewords have these lengths. Moreover, for all $1 \leq i \leq n$,

$$\log_b \frac{1}{p_i} \leq l_i < \log_b \frac{1}{p_i} + 1.$$

And from this we conclude (by first multiplying through by p_i followed by summing over i)

$$H_b(p_1, \ldots, p_n) \leq \text{MinAveLen}(P) \leq \text{AveLen}(C, P) < H_b(p_1, \ldots, p_n) + 1.$$

□

4.1 Improving the information rate

The importance of the MinAveLen(P) value is that it represents the average number of codeword symbols that must be transmitted in order to communicate a single symbol/event of the information source. We thus refer to the ratio MinAveLen(P)/$H_b(P)$ as the **information transmission rate (ITR)**, as it gives the average number of symbols transmitted per average amount of information conveyed.

Given an information source $S = (\mathcal{A}, P = \{p_1, \ldots, p_n\})$, the Noiseless Coding Theorem guarantees that the average number of transmitted symbols can be made better than $H_b(p_1, \ldots, p_n) + 1$. We now show how the average can be made arbitrarily close to $H_b(p_1, \ldots, p_n)$, showing that the ITR can be made to approach 1. The key idea is to use the m-ary extension function

$$f^m : \mathcal{A}^m \to \mathcal{C},$$

as the encoding function and use a Huffman code to encode each m-ary word. Moreover, if we assume that the symbols are occurring independently of one another, then the average information conveyed by an m-ary word is $mH_b(p_1, \ldots, p_n)$. Now, if we let P^m denote the probability distribution for a sequence of m independent symbols, Then for a single transmission of m symbols we have,

$$mH_b(p_1, \ldots, p_n) \leq \text{MinAveLen}(P^m) < mH_b(p_1, \ldots, p_n) + 1,$$

and dividing by m we see that

$$H_b(p_1, \ldots, p_n) \leq \frac{\text{MinAveLen}(P^m)}{m} < H_b(p_1, \ldots, p_n) + \frac{1}{m}.$$

But, since MinAveLen(P^m) gives the average encoding length for m symbols,

$$\frac{\text{MinAveLen}(P^m)}{m}$$

gives the average encoding length for a single symbol, and the average converges to $H_b(p_1, \ldots, p_n)$ for increasing values of m. Therefore, the ITR can be made arbitrarily close to 1 via increasing values of m.

The following example is taken from Roman's "Information and Coding Theory" (see page 65).

Example 4.4. Given the probability distibution $P = (\frac{1}{4}, \frac{3}{4})$, its Huffman code $\{0, 1\}$ has an average length equal to 1, while $H(\frac{1}{4}, \frac{3}{4}) = 0.811$. Determine the average codeword length for both a binary extension and a ternary extension.

Solution.

References

1. T. Cover, J. Thomas, "Elements of Information Theory", 2nd Edition, Wiley-Interscience, 2006
2. L. Mlodinow, "The Drunkard's Walk: How Randomness Rules our Lives", Vintage Press, 2009
3. S. Roman, "Information and Coding Theory", Springer-Verlag, 1992
4. S. Ross, "Introduction to Probability Models", 10th Edition, Academic Press, 2009