**Shannon Entropy: Source Coding Problem**[*] 2025-01-20

jessekelighine.com
Jesse C. Chen 陳捷

## 1 Motivation

Let $\mathcal{X}$ be a finite sample space, and let $X$ be a random variable with $\mathbf{P}$ as the induced probability measure on $\mathcal{X}$. How much *information* does $X$ carry? We can approach this question as follows: If we encode each possible outcome in $\mathcal{X}$ using a given set of symbols, on average, what is the number of symbols required to represent a realization of $X$?

**Example 1.** Suppose we have a rigged six-sided die with the following probabilities and encoding in binary (two symbols):

| Face | Probability | Encoding |
|---|---|---|
| ⚀ | 80% | 1 |
| ⚁ | 10% | 01 |
| ⚂ | 2.5% | 0011 |
| ⚃ | 2.5% | 0010 |
| ⚄ | 2.5% | 0001 |
| ⚅ | 2.5% | 0000 |

Since it is highly probable that ⚀ will occur, it is efficient to assign it a short codeword. Under this scheme, 1.4 bits is expected to encode an outcome since

$$0.8 \times 1 \text{ bit} + 0.1 \times 2 \text{ bits} + (0.025 \times 4 \text{ bits}) \times 4 = 1.4 \text{ bits}.$$

If the die is so rigged that only ⚀ ever occurs, then the die carries no information, as the outcome is entirely predictable. In this case, the required encoding length would be 0 bits.

In the following sections, we will formalize this idea of information through encoding and derive Shannon entropy as a quantitative measure of information.

## 2 Code Function

**Definition 1** ($d$-ary Code Function)**.** A *$d$-ary code function* $\mathsf{C}$ maps $x \in \mathcal{X}$ to a finite string, denoted by $\mathsf{C}(x)$, consisting of $d$ distinct symbols. The output of $\mathsf{C}$ is referred to as the *codeword*. Codewords can have different lengths and let $\ell_{\mathsf{C}}(x)$ denote the length of the codeword used to encode $x \in \mathcal{X}$.

**Remark 1.** For any $d$-ary code function $\mathsf{C}$ to be considered reasonable, it must allow the encoded message to be decoded unambiguously. This requires certain properties to hold, as defined below.

**Definition 2** (Non-singular)**.** A $d$-ary code function $\mathsf{C}$ is *non-singular* if for all $x, x' \in \mathcal{X}$, we have $\mathsf{C}(x) = \mathsf{C}(x')$ iff $x = x'$.

[*]This introduction draws heavily from lecture note Duchi (2024), Chapter 2.

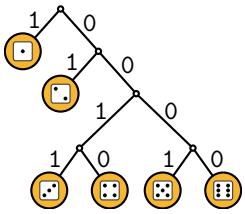


Figure 1: Prefix Tree: Dice.

**Definition 3** (Uniquely Decodable)**.** A $d$-ary code function $\mathsf{C}$ is *uniquely decodable* if for all sequences $x_1, ..., x_n \in \mathcal{X}$ and $x'_1, ..., x'_n \in \mathcal{X}$, we have

$$\mathsf{C}(x_1) \cdots \mathsf{C}(x_n) = \mathsf{C}(x'_1) \cdots \mathsf{C}(x'_n) \quad \text{iff} \quad x_1 = x'_1, ..., x_n = x'_n.$$

That is, there is no ambiguity in any encoded sequences.

**Remark 2.** If a code function is uniquely decodable, then it must be non-singular, but not the other way round. A particularly useful type of uniquely decodable code is the *prefix code*.

## 3 Prefix Code

**Definition 4** (Prefix Code)**.** A $d$-ary code function $\mathsf{C}$ is said to be a *prefix code* if $\nexists c, c' \in \{\mathsf{C}(x)\}_{x \in \mathcal{X}}$ such that $c'$ starts with $c$. That is, no whole codeword is a prefix of any other codeword.

**Remark 3.** Since no codeword is the start of any other codeword, a prefix code can be represented by a graph called *prefix tree*. And there is a one-to-one correspondence between prefix trees and prefix codes. The encoding given in Example 1 is a prefix code and the corresponding prefix tree is shown in Figure 1.

**Lemma 1.** Any prefix code is uniquely decodable.

*Proof.* This should be pretty obvious with the prefix tree representation. ∎

**Remark 4.** Prefix codes are among the simplest uniquely decodable codes to work with. While there are other uniquely decodable encoding schemes, prefix codes are special because of the Kraft-McMillan Inequality.

**Theorem 1** (Kraft-McMillan Inequality)**.** Let $\mathcal{X}$ be a finite set, and let $\ell : \mathcal{X} \to \mathbb{N}$ be a function. If $\ell(x)$ is the length of the encoding of the element $x$ in a uniquely decodable $d$-ary, then

$$\sum_{x \in \mathcal{X}} d^{-\ell(x)} \leq 1.$$

Conversely, given any function $\ell : \mathcal{X} \to \mathbb{N}$ that satisfies the inequality above, there exists a prefix code $\mathsf{C}$ such that $\ell_{\mathsf{C}}(x) = \ell(x)\ \forall x \in \mathcal{X}$.

*Proof.* First suppose that $|\mathcal{X}| < \infty$. Let $\ell_{\max} \coloneqq \max_{x\in\mathcal{X}} \ell(x)$ denote the longest codeword. Let $x_{1:n} \coloneqq (x_1, ..., x_n) \in \mathcal{X}^n$. Define the notation

$$E_n(m) \coloneqq \{x_{1:n} : \ell(x_{1:n}) = m\}$$

where $\ell(x_{1:n}) = \sum_{i=1}^n \ell(x_i)$. That is, $E_n(m)$ denotes the set of strings of $n$ elements in $\mathcal{X}$ that is encoded with $m$ symbols. Clearly, we have $\ell(x_{1:n}) \leq n\ell_{\max}$, and by unique decodability, we must have $|E_n(m)| \leq d^m$. Consider the following rewriting of the sum for any fixed $n$:

$$\sum_{x_{1:n}\in\mathcal{X}^n} d^{-\ell(x_{1:n})} = \sum_{m=1}^{n\ell_{\max}} |E_n(m)|d^{-m} \leq \sum_{m=1}^{n\ell_{\max}} d^m d^{-m} = n\ell_{\max}.$$

We can rewrite the sum in another way:

$$\sum_{x_{1:n}\in\mathcal{X}^n} d^{-\ell(x_{1:n})} = \sum_{x_{1:n}\in\mathcal{X}^n} d^{-\ell(x_1)} \cdots d^{-\ell(x_n)} = \left(\sum_{x\in\mathcal{X}} d^{-\ell(x)}\right)^n.$$

Hence, for all $n$ we have

$$\sum_{x\in\mathcal{X}} d^{-\ell(x)} = \left(\sum_{x_{1:n}\in\mathcal{X}^n} d^{-\ell(x_{1:n})}\right)^{1/n} \leq (n\ell_{\max})^{1/n}.$$

Taking $\inf_{n\in\mathbb{N}}$ on both sides and we obtain the inequality.

Now consider the converse. WLOG, let elements of $\mathcal{X}$ be a consecutive subset of $\mathbb{N}$ such that $\ell(x) \leq \ell(y)\ \forall x \leq y$ starting from 1. We place all the elements in a $d$-ary prefix tree as follows: First, place $1 \in \mathcal{X}$ at a node of depth $\ell(1)$ and prune all the descendants from that node. This means that for an arbitrary $n \in \mathcal{X}$, there are $d^{\ell(n)-\ell(1)}$ nodes of depth $\ell(n)$ removed from the pruning. Then, we place $2 \in \mathcal{X}$ at a node of depth $\ell(2)$, removing $d^{\ell(n)-\ell(2)}$ nodes of depth $\ell(n)$ from the prefix tree. This process continues until $n \in \mathcal{X}$ is placed in the tree. Note that this process never removes more nodes than there are available to us at any point $n$ since the Kraft-McMillan Inequality holds:

$$\sum_{i=1}^n d^{\ell(n)-\ell(i)} = d^{\ell(n)} \sum_{i=1}^n d^{-\ell(i)} \leq d^{\ell(n)}.$$

Therefore, a prefix tree, hence a prefix code, can always be constructed by this method. ∎

**Remark 5.** The Kraft-McMillan Inequality characterizes unique decodability. Furthermore, it shows that we can always re-encode a uniquely decodable code function using a prefix code.

## 4 Entropy as Information

**Definition 5** (Shannon Entropy)**.** Let $X$ be a random variable with sample space $\mathcal{X}$. Define *Shannon entropy* as

$$\mathrm{H}_d(X) \coloneqq -\sum_{x\in\mathcal{X}} \mathbf{P}(x) \log_d \mathbf{P}(x)$$

where $\mathbf{P}$ is the probability measure induced by $X$ on $\mathcal{X}$.

**Remark 6.** Note the negative sign in $\mathrm{H}_d(X)$. Since $\log \mathbf{P}(x)$ is always non-positive, the negative sign keeps the entire thing non-negative. And by convention, for any $x \in \mathcal{X}$ such that $\mathbf{P}(x) = 0$, we define $\mathbf{P}(x) \log_d \mathbf{P}(x) = 0$.

**Remark 7.** It turns out that the minimal average length of a codeword necessary to encode $X$ is given by $\mathrm{H}_d(X)$. This result is called Shannon's Source Coding Theorem.

**Theorem 2** (Shannon's Source Coding Theorem)**.** Let $X$ be a random variable with sample space $\mathcal{X}$. Let $\mathcal{C}$ denote the set of all possible uniquely decodable $d$-ary code functions for $\mathcal{X}$. Then

$$\mathrm{H}_d(X) \leq \inf_{\mathsf{C} \in \mathcal{C}} \mathbf{E}\, \ell_{\mathsf{C}}(X) \leq \mathrm{H}_d(X) + 1$$

where $\mathbf{P}$ is the probability measure induced by $X$ on $\mathcal{X}$ and $\mathbf{E}$ denotes the expectation operator.

*Proof.* By Kraft-McMillan Inequality, solving $\inf_{\mathsf{C} \in \mathcal{C}} \mathbf{E}\, \ell_{\mathsf{C}}(X)$ is equivalent to solving

$$\inf_{\ell} \sum_{x \in \mathcal{X}} \mathbf{P}(x) \ell(x) \quad \text{subject to} \quad \sum_{x \in \mathcal{X}} d^{-\ell(x)} \leq 1.$$

This can be done by setting up and solving the Lagrangian:

$$\begin{aligned} \mathcal{L} = \sum_{x \in \mathcal{X}} \mathbf{P}(x) \ell(x) + \lambda \left( 1 - \sum_{x \in \mathcal{X}} d^{-\ell(x)} \right) &\implies \mathbf{P}(x) - \lambda d^{-\ell(x)} \log(d) \overset{\text{let}}{=} 0 \ \forall x \\ &\implies \ell(x) = -\log_d \mathbf{P}(x). \end{aligned}$$

Thus, the lower bound is obtained. For the upper bound, let $\ell(x) = \lceil -\log_d \mathbf{P}(x) \rceil$. Note that this $\ell$ satisfies Kraft-McMillan Inequality, hence a prefix code can be constructed. Therefore, we have

$$\mathbf{E}\, \ell(X) = \sum_{x \in \mathcal{X}} \mathbf{P}(x) \lceil -\log_d \mathbf{P}(x) \rceil \leq \sum_{x \in \mathcal{X}} \mathbf{P}(x)(-\log_d \mathbf{P}(x)) + 1 = \mathrm{H}_d(X) + 1,$$

obtaining the upper bound. ∎

**Remark 8.** For the rigged die in Example 1 the entropy is about 1.12 bits. Our encoding scheme achieves an expected length of 1.4 bits, which is close to the theoretical limit. There are also an asymptotic version of this theorem: If we observe a stationary sequence $X_1, X_2, ...$, then we can re-encode the $X_i$'s in blocks. This way the expected length of encoding can be arbitrarily close to its entropy. One can refer to Duchi (2024), Chapter 2 for the proof.

## References

Duchi, John (2024). *Lecture Notes on Statistics and Information Theory*. URL: https://web.stanford.edu/class/stats311/lecture-notes.pdf.