Complexity and Information Theory

Pietro Marcatti

First Semester 2022/2023

1 Introduction

The information comunicated with a sentence, or any other way, is always dependant on the context. Same goes for the quality for the information. If an event has a low probability of occurring in a given context the information it provides is high, inversely if the probability is high the information is little.
We could try to describe information with a simple inverse model:

$$Information = \frac{1}{p(E)}, \quad \text{E=event}$$

$$p(E) \rightsquigarrow 0 \quad Information \rightsquigarrow \infty$$

$$p(E) \rightsquigarrow 1 \quad Information \rightsquigarrow 1$$

Instead if we were to adopt a logaritmic model:

$$Information = \log \frac{1}{p(E)} = -\log p(E), \quad \text{E=event}$$

$$p(E) \rightsquigarrow 0 \quad Information \rightsquigarrow \infty$$

$$p(E) \rightsquigarrow 1 \quad Information \rightsquigarrow 0$$

Let's consider an alphabet as a group of possible events for which we have a probability distribution:

$$A = a_1, a_2, a_3, \ldots, a_k$$

$$P = p_1, p_2, p_3, \ldots, p_k$$

meaning that p_i is the probability to observe the event a_1. We can then calculate the average quantity of information as follows:

$$\sum_{i=1}^{k} p_i \cdot (-\log p_i) = \underbrace{\mathbb{H}(P)}_{ShannonEntropy} \tag{1}$$

1.1 Coin Flip Example

Fair Coin

$$A = \{H, T\}$$

$$P(H) = \frac{1}{2}, \quad P(T) = \frac{1}{2}$$

$$\mathbb{H}(P) = \frac{1}{2} \cdot (-\log \frac{1}{2}) + \frac{1}{2} \cdot (-\log \frac{1}{2}) = 1$$

Every time we filp the fair coin we expect to gain a bit of information. Unfari Coin

$$A = \{H, T\}$$

$$P(H) = \frac{9}{10}, \quad P(T) = \frac{1}{10}$$

$$\mathbb{H}(P) = \frac{9}{10} \cdot (-\log \frac{9}{10}) + \frac{1}{10} \cdot (-\log \frac{1}{10}) \rightsquigarrow 0.32$$

1.2 Properties of Entropy

- $\mathbb{H}(P) \geq 0$
- $\mathbb{H}(P) = 0$ if there is an event with probability 1
- $\mathbb{H}$ is continuous with respect to P
- If an event gets split the entropy should be additive
- $\mathbb{H}(p_1, p_2, \ldots, p_k) \leq \log k = \mathbb{H}(\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k})$

To prove the last property we can first show that for a equi-distribution we get $\mathbb{H} = \log k$. To prove that this is also the maximum we use Jensen disequality:

$$\forall f, f''(x) < 0 \quad in \; [a, b], (a, b) \in \mathbb{R}^2$$

$$f\Big(\sum_{i=1}^{k} \lambda_i \cdot x_i\Big) \geq \sum_{i=1}^{k} \lambda_i \cdot f(x_i), \quad \sum \lambda_i = 1$$

2 Data Compression

Before we can talk about data compression it is necessary to explore the meaning and the functiong of encodings. Given two alphabets $A = a_1, a_2, \ldots, a_k$ and $B = b_1, b_2, \ldots, b_k$ an encoding is function:

$$\varphi : A^* \longrightarrow B^* \tag{2}$$

For a function to be a suitable encoding it must be at least an injective function (one-way function). In information theory and in data encoding in particular we refer to injective encoding as **uniquely decodable** encodings. Prefix codes are uniquely decodable codes that can be decoded without delay and are written in the form:

$$A = \{a_1, a_2, \ldots, a_k\} \xrightarrow{\varphi} B = \{b_1, b_2, \ldots, b_D\} \tag{3}$$

where $\varphi_i = |\varphi(a_i)|$

2.1 Kraft-McMillen 1958 Inverse Theorem

If φ is a uniquely decodable code, then:

$$\sum_{i=1}^{k} D^{1-l_i} <= 1 \tag{4}$$

D is the cardinality of the output alphabet, l_i is the lenght of the encoding for a_i. We can quickly see that, there can't be an uniquely decodable code that uses three encodings of length 1 to map over the binary alphabet:

$$A = \{a, b, c\}, \quad B = \{0, 1\}, \quad l_a = l_b = l_c = 1 \longrightarrow \frac{1}{2} + \frac{1}{2} + \frac{1}{2} > 1 \tag{5}$$

Instead, if we were to insist on using a length 10 encoding for everyone of the input characters:

$$A = \{a, b, c\}, \quad B = \{0, 1\}, \quad l_a = l_b = l_c = 10 \longrightarrow \frac{1}{2^{10}} + \frac{1}{2^{10}} + \frac{1}{2^{10}} << 1 \tag{6}$$

2.1.1 Proof 1

Assume the code is uniquely decodable and it is a prefix code, we can sort the input alphabet so that the length of the encoding satisfies the following:

$$a_1 \leq a_2 \leq \ldots \leq a_k$$

A character with an enconding of length i generates a sub-tree rooted in a_i with height $l - l_i$. This sub-tree contains $D^(l - l_i)$ nodes that cannot be used in the encoding because they would share an encoded prefix.

Insert diagram
In the original three the following number of leaves

$$D^l - \sum_{i=1}^{k-1} D^{l-l_1}$$

The $k-1$ term that bounds the sum is needed because we need to make sure that one last leaf is available to assign it to the last character a_k.

$$\begin{aligned}
D^l - \textstyle\sum_{i=1}^{k-1} D^{l-l_i} &\geq 1 \\
&\qquad \textit{Group by } D^l \\
D^l(1 - \textstyle\sum_{i=1}^{k-1} D^{-l_i}) &\geq 1 \\
&\qquad \textit{Divide by } D^l \\
1 - \textstyle\sum_{i=1}^{k-1} D^{-l_i} &\geq D^{-l} \\
&\qquad D^{-l} \textit{ is the last term of the sum} \\
1 &\geq \textstyle\sum_{i=1}^{k} D^{-l_i}
\end{aligned}$$

2.1.2 Proof 2 - General case

Let's define $N(n,h)$ the number of strings of alphabet A^n having encoding length h. It must be true that $N(n,h) \leq D^h$ because φ is uniquely decodable.

$$D^{-l_1} + D^{-l_2} + \ldots + D^{-l_k} \leq 1 \tag{7}$$

$\forall n, n \in \mathbb{N}$ let's consider the object $(D^{-l_1} + D^{-l_2} + \ldots + D^{-l_k})^n$. If written as a product it takes the following form:

$$D^{-l_1 \cdot n} + D^{-l_1 \cdot (n-1) - l_2} + \ldots + D^{-l_k \cdot n}$$

As n tends towards infinity the exponent will behave differently depending on the value of the base. If the base is < 1 it will be bound between 0 and 1, and in any case < 1. If the base is > 1 it will grow towards infinity faster than any polynomial function. Lastly, if the base is $= 1$ it will remain constant. To prove the theorem we will demonstrate that the object is dominated by a linear function, meaning that it cannot be in the second case. All of the members' exponents follow this chain of disequalities: $l_1 \cdot n \leq exp \leq l_k \cdot n$. That is, because we sorted the character by encoding length, 1 being the shortest while k the longest.

$$\begin{aligned}
\underset{prob.0}{N(n,1)} D^{-1} + \ldots + N(n, l_1 \cdot n) D^{l_1 \cdot n} + \ldots + N(n, l_k \cdot n) D^{-l_k \cdot n} \\
\leq D^1 D^{-1} + \ldots + D_k^l D^{-l_k} \leq l_k \cdot n
\end{aligned}$$

Since the object is dominated by $l_k \cdot n$ which is linear, it must mean that it also grows at most linearly, meaning that the base of the exponent must be ≤ 1.

2.2 Kraft-McMillen Direct Theorem

If given $l_1, l_2, \ldots, l_k$ and D such that $\sum_{i=1}^{k} D^{-l_i} \leq 1$ there must exist a prefix code φ having $l_1, l_2, \ldots, l_k$ as lengths of the encoding.

2.2.1 Proof

Proof can be given by construction by producing the complete D-ary tree

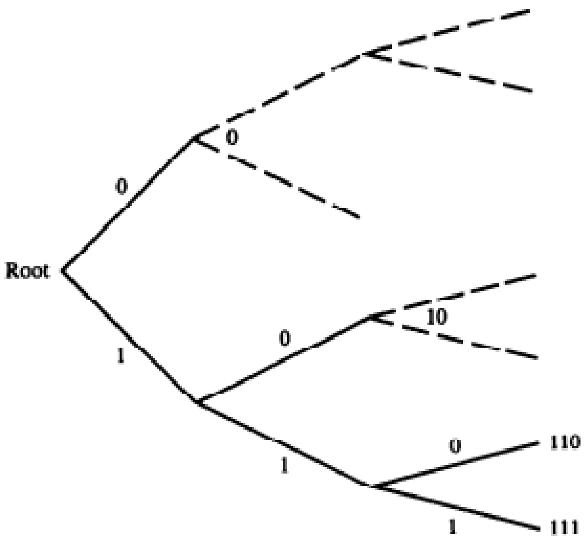

Figure 5.2. Code tree for the Kraft inequality.

Observation: Prefix codes compress as good as any other uniquely decodable code.

2.2.2 Average expected code length - Definition

Given the input alphabet A, P the probability and B the output we define the Expected Length as:

$$EL(\varphi) = \sum_{i=1}^{k} p_i \cdot l_i = \sum_{i=1}^{k} p_i |\varphi(a_i)| \tag{8}$$

Our goal is to find prefix codes that minimize EL.

2.3 1st Shannon Theorem 1948

If φ is uniquely decodable code then $EL(\varphi) \geq \mathbb{H}_D(P) = \sum_{i=1}^{k} p_i \cdot log_D p_i$ The proof relies on Kraft-McMillen theorem and this simple logarithmic property:

$$log_e x \leq x - 1, \quad -log_e x \geq 1 - x$$

2.3.1 Proof

$$
\begin{aligned}
& EL(\varphi) - \mathbb{H}_D(P) = \textstyle\sum_{i=1}^{k} p_i \cdot l_i + \sum_{i=1}^{k} p_i \cdot log_D p_i && \textit{Group by } p_i \textit{ and force } l_i \textit{ as } log_D \\
& = \textstyle\sum_{i=1}^{k} p_i \cdot log_D (D^{l_i} p_i) && \textit{Change to base e, take out the common term} \\
& = \tfrac{1}{log_e D} \cdot \textstyle\sum_{i=1}^{k} p_i \cdot log_e (D^{l_i} p_i) && \textit{Group by } -1 \\
& = \tfrac{-1}{log_e D} \cdot \textstyle\sum_{i=1}^{k} p_i \cdot log_e \tfrac{1}{D^{l_i} p_i} && \textit{Using the above property} \\
& \geq \tfrac{-1}{log_e D} \cdot \textstyle\sum_{i=1}^{k} p_i \cdot (\tfrac{1}{D^{l_i} p_i} - 1) \\
& = \tfrac{-1}{log_e D} \cdot \left(\textstyle\sum_{i=1}^{k} \tfrac{1}{D^{l_i}} - \underbrace{\cancel{\textstyle\sum_{i=1}^{k} p_i}}_{=1} \right) && \textit{Group by } -1 \\
& = \tfrac{1}{log_e D} \underbrace{\left(1 - \textstyle\sum_{i=1}^{k} \tfrac{1}{D^{l_i}} \right)}_{KMM \longrightarrow \geq 0} \geq 0
\end{aligned}
$$

An optimum code achieves the equality between the expected lenght and the entropy of the distribution.

Shannon Code

$$EL(\varphi) = \sum_{i=1}^{k} p_i \cdot l_i$$

$$\mathbb{H}_D(P) = -\sum_{i=1}^{k} p_i \cdot \log_D \frac{1}{p_i}$$

First observation: $\log_D \frac{1}{p_i} \in \mathbb{R}, \quad l_i \in \mathbb{N} \longrightarrow l_i = \lceil \log_D \frac{1}{p_i} \rceil$
Second observation: We can use the Direct Theorem to prove φ exists
Example:

$$\sum_{i=1}^{k} D^{-\lceil \log_D \frac{1}{p_i} \rceil} \leq 1$$

$$\lceil \log_D \frac{1}{p_i} \rceil = \log_D \frac{1}{p_i} + \beta_i \quad 0 \leq \beta_i < 1$$

$$\sum_{i=1}^{k} D^{\log_D p_i - \beta_i} = \sum_{i=1}^{k} D^{\log_D p_i} \cdot \frac{1}{D^{\beta_i}} = \sum_{i=1}^{k} p_i \cdot \frac{1}{D^{\beta_i}}$$

Example: $A = \{a, b\}$ $B = \{0, 1\}$ $P = \left\{1 - \frac{1}{32}, \frac{1}{32}\right\}$ $l_b = \log_2 2^5 = 5$ $\lceil \log_2 1 - \frac{1}{32} \rceil = 1 = l_a$ But this is not efficient because we are assigning unnecessarily long codes to characters with low probability.

Shannon-Fano Code

$a_1, a_2, \ldots, a_k \longrightarrow p_1 \leq p_2 \ldots \leq p_k$ The aim is to balance the sum of the probabilities assigned to the branches of the root. $|\sum_{i=1}^{h} p_i - \sum_{i=h+1}^{k} p_i|$. To obtain balance we try to minimize this absolute value. We then repeat the calculation recursively. Example: $A = \{a, b, c, d, e, f\}$ $\quad P = \left\{ \underbrace{\frac{40}{100}, \frac{18}{100}}_{\frac{58}{100}}, \underbrace{\frac{15}{100}, \frac{13}{100}, \frac{10}{100}, \frac{4}{100}}_{\frac{42}{100}} \right\}$

Fai l'esercizio e disegna l'albero. Why is Shannon-Fano compromising and not splitting exactly in half or as best as possible? Because the problem of splitting a set in equal subsets is NP-Complete.

We can demonstrate that Shannon Codes are suboptimal:

$$\mathbb{H}_D(P) \leq \underset{\sum_{i=1}^{k} p_i \cdot \log_D \frac{1}{p_i} + \sum_{i=1}^{k} p_i \cdot \beta_i}{EL(\varphi)} \leq \mathbb{H}_D(P) + 1$$

$\sum_{i=1}^{k} p_i \cdot \beta_i \leq 1$

Shannon on strings of lenght n over input alphabet

Example: $A = \{a, b\}$ $B = \{0, 1\}$ $P = \left\{\frac{3}{4}\right\}$ $\mathbb{H}_2(P) = 0.81$ $EL(\varphi) = 1.25$ $\quad Eff. = \frac{\mathbb{H}_D(P)}{EL(\varphi)} \leq 1$ $Eff. = \frac{\mathbb{H}_2(P)}{EL(\varphi)} = 0.64$

Let's try to calculate the efficiency now with pairs $A' = \{aa, ab, ba, bb\}$ $P = \{9/16, 3/16, 3/16, 1/16\} = P^2$ $\mathbb{H}_D(P') = 1.62 = \mathbb{H}_D(P^2) = 2 \cdot \mathbb{H}_D(P)$ $l_{aa} = \lceil \log_2 16/9 \rceil = 1$ $EL(\varphi) = 1.93 Eff. = 0.83$ If we pretend we want to send a message 1000 characters long, with the single character encoding we expect a message lenght of 1.25k characters, Instead with the pairs encoding we expect $1.93/2 = \rightsquigarrow 0.97$k characters.

property: given two events x, y independent $\mathbb{H}(xory) = H(x) + H(Y)$ Proof:

$$H(xorY) = \sum_{i,j} p_i q_j \log p_i q_j = -\sum_{i,j} p_i q_j \log p_i - \sum_{i,j} p_i q_j \log q_j$$

$$= -\sum_j q_j \cdot \sum_i p_i \log p_i - \sum_i p_i \cdot \sum_j q_j \log q_j = \mathbb{H}(x) + \mathbb{H}(y)$$

$EL(\varphi_n) \geq \mathbb{H}_D(P^n) = n \cdot \mathbb{H}_D(P)$ For Shannon Codes (and all sub-optimal codes) we have:

$$n \cdot \mathbb{H}_D(P) \leq EL(\varphi_n) < n \cdot \mathbb{H}_D(P) + 1$$

If we want to reason in terms of a single input alphabet value we divide the disequality by n. We can see that for $n \rightsquigarrow \infty$ the expected lenght is squeezed between the terms of the disequality.

Huffman Codes

$A = \{a_1, a_2, \ldots, a_k\}$, $p_1 \leq p_2 \ldots \leq p_k$ We start by taking the two leasts probable characters and we put them at the leaves, creating a phantom character with probability the sum of the two. I keep repeating this process until we obtain a complete tree.