5. Source Coding: Lossless Compression

Recall Shannon's entropy (information entropy)

How Much Compression Is Possible?

Suppose that we're faced with N equally probable choices and we receive information that narrows it down to M choices. Shannon offered the following formula for the information received:

$$\log_2(N/M) \text{ bits of information}$$

▼ Example

one flip of a fair coin:

Before the flip, there are two equally probable choices: heads or tails. After the flip, we've narrowed it down to one choice. Amount of information $= \log_2(2/1) = 1\ bits$.

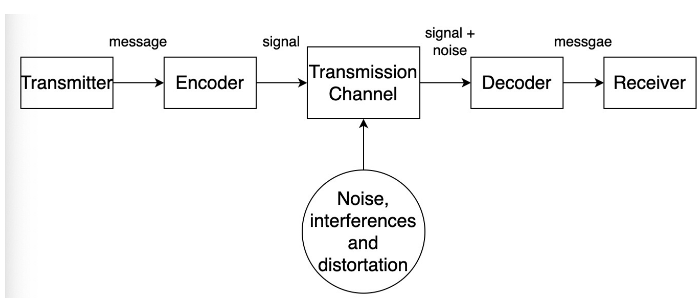
We can use this equation to compute the information content when learning of a choice by computing the weighted average of the information received for each particular choice:

$$\text{Information content in a choice} = \sum_{i=1}^{N} p_i log_2(1/p_i)$$

The **main problem** in source coding is to ensure that the most probable source symbols are represented by the shortest codewords, and the less probable by longer codewords as necessary, the weighted average codeword length being minimized within the bounds of Kraft's inequality.

▼ Source coding

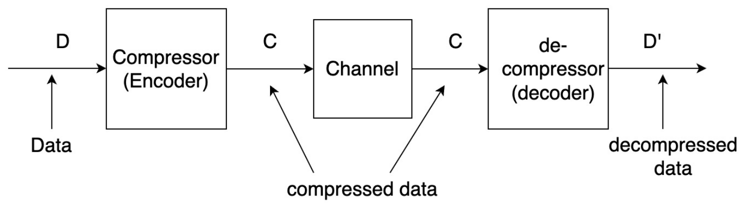
Architecture of Communication Systems



Goal: Removing uncertainty and gaining information

Source Coding: To remove redundancy in the information to make the message smaller.

Source coding = Data compression



The goal: $size(C) < size(D)$

Compression Ratio: $\rho = \frac{\text{size}(C)}{\text{size}(D)} < 1$

Channel:

- Communications channel ("from here to there")
- Storage channel ("from now to then")

Source Code: A mapping C from a source alphabet $\mathcal{X}$ to $D-\text{ary}$ sequences

- $\mathcal{D}^*$ is set of finite-length strings of symbols from $D-\text{ary}$ alphabet $\mathcal{D} = \{1, 2, .., D\}$, i.e. $\mathcal{D}^* = \mathcal{D} \cup \mathcal{D}^2 \cup \mathcal{D}^3 \cup$ .....
- $C(x) \in \mathcal{D}^*$ is the codeword for $x \in \mathcal{X}$.
- $\ell(x)$ is the length of $C(x)$.

A code is **nonsingular** if

$$x \neq \tilde{x} \Rightarrow C(x) \neq C(\tilde{x})$$

The **extension** of the code C is the mapping from finite length strings of $\mathcal{X}$ to finite length strings of $\mathcal{D}$

$$C(\underbrace{x_1 x_2 \cdots x_n}_{\text{input (source)}}) = \underbrace{C\left(x_1\right) C\left(x_2\right) \cdots C\left(x_n\right)}_{\text{output (code)}}$$

A code C is **uniquely decodable** if its extension C^* is nonsingular, i.e., for all n,

$$x_1 x_2 \cdots x_n \neq \tilde{x}_1 \tilde{x}_2 \cdots \tilde{x}_n \quad \Longrightarrow \quad C\left(x_1\right), C\left(x_2\right), \cdots, C\left(x_n\right) \neq C\left(\tilde{x}_1\right), C\left(\tilde{x}_2\right), \cdots, C\left(\tilde{x}_n\right)$$

A code is **prefix-free** if no codeword is prefixed by another codeword. Such codes are also known as "prefix" codes or instantaneous codes.

Given a distribution $p(x)$ on the input symbol, the goal is to minimize the expected length per-symbol

$$\mathbb{E}[\ell(X)] = \sum_{x\in\mathcal{X}} \ell(x)p(x)$$

▼ Example:

prefix code is such that no codeword is a prefix of any other codewords. For instance, suppose we'd like to encode the letters $\{a, b, c, d\}$ in binaries, i.e. using {0,1}. Then one possible coding scheme is as follows:

$$\begin{cases} a \to 0 \\ b \to 01 \\ c \to 011 \\ d \to 0111 \end{cases}$$

This is uniquely decodable but it is not a prefix code because the codeword for a is a prefix for the codeword for b. This means that we can not instantaneously decode a without waiting for the next bit of data (to determine whether it is actually a or just the first half of b.)

Alternatively, we can encode using a prefix code as follows:

$$\begin{cases} a \to 0 \\ b \to 10 \\ c \to 110 \\ d \to 111 \end{cases}$$

As you can see no codeword is a prefix of another codeword.

▼ Compression

Shannon Source Coding Theorem: For any source distribution $p(x)$, the expected length $\mathbb{E}[\ell(X)]$ of the optimal
uniquely decodable $D-ary$ code obeys

$$\frac{H(X)}{\log D} \le \mathbb{E}[\ell(X)] \le \frac{H(X)}{\log D} + 1$$

Furthermore, there exists a prefix-free code which is optimal.

Let $\ell(x)$be the length function associated with a code C. A code C satisfies the **Kraft Inequality** if and only if

$$\sum_{x\in\mathcal{X}} D^{-\ell(x)} \le 1 \qquad \text{(Kraft Inequality)}$$

▼ Example

If we encoding letters $\{a, b, c, d\}$ in binaries as following:

$$\begin{cases} a \to 0 \\ b \to 10 \\ c \to 110 \\ d \to 111 \end{cases}$$

Followed by Kraft Inequality, we have $D = 2$ and then

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} \leq 1$$

Theorem: Every uniquely decodable code satisfies the Kraft inequality

▼ Proof

1. Let C be a uniquely decodable source code.
2. For a source sequence x^n, the length of the extended codeword $C(x^n)$ is given by

$$\ell(x^n) = \sum_{i=1}^{n} \ell(x_i)$$

   and thus, the length function of the extended codeword obeys

$$n\ell(x)_{min} \leq \ell(x^n) \leq n\ell_{max}$$

3. Let A_k be the number of source sequences of length n for which $\ell(x^n) = k$, i.e.

$$A_k = \#\{x^n \in \mathcal{X}^n \ : \ \ell(x^n) = k\}$$

4. Since the code is uniquely decodable, the number of source sequence with codewords of length k cannot exceed the number of $D-ary$ sequences of length k , an so

$$A_k \leq D^k$$

5. The extended codeword lengths must obey

$$\begin{aligned} \sum_{x^n \in \mathcal{X}^n} D^{-\ell(x^n)} &= \sum_{x^n \in \mathcal{X}^n} \left(\sum_{k=1}^{\infty} \mathbf{1}\left(\ell\left(x^n\right) = k\right) D^{-k} \right) \\ &= \sum_{k=1}^{\infty} \left(\sum_{x^n \in \mathcal{X}^n} \mathbf{1}\left(\ell\left(x^n\right) = k\right) \right) D^{-k} \\ &= \sum_{k=1}^{\infty} A_k D^{-k} \\ &\leq \sum_{k=n\ell_{\min}}^{n\ell_{\max}} D^k D^{-k} \\ &\leq n\ell_{\max} \end{aligned}$$

6. The extended codeword lengths must also obey

$$\sum_{x^n \in \mathcal{X}^n} D^{-\ell(x^n)} = \sum_{x_1 \in \mathcal{X}} D^{-\ell(x_1)} \sum_{x_2 \in \mathcal{X}} D^{-\ell(x_2)} \times \cdots \times \sum_{x_n \in \mathcal{X}} D^{-\ell(x_n)} = \left[\sum_{x \in \mathcal{X}} D^{-\ell(x)}\right]^n$$

7. Combining the above displays shows that

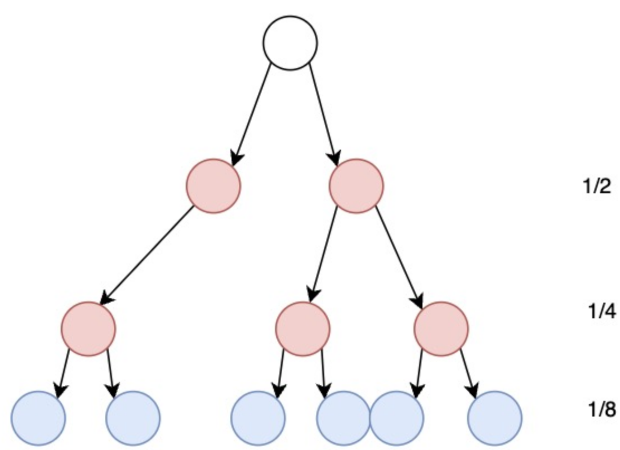
$$\left[\sum_{x \in \mathcal{X}} D^{-\ell(x)}\right]^n \leq n\ell_{\max} \quad \text{for all } n$$

8. If the code does not satisfy the Kraft inequality, then the left hand side will blow up exponentially as n becomes large, and this inequality will be violated. Thus, the code must satisfy the Kraft inequality.

▼ Codes on Tree

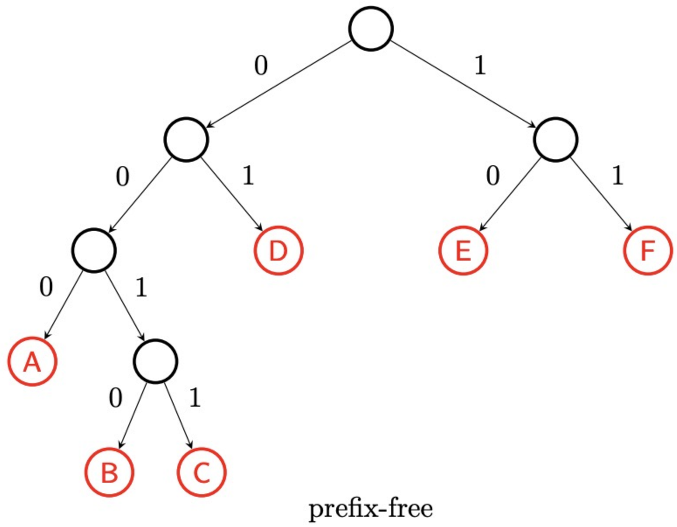
Any D-ary code can be represented as a D-ary tree, A D-ary tree consists of a root with branches, nodes, and leaves. The root and every node has exactly D children.

For example, for a binary tree, here $D = 2$.



From this figure we can know that the sum of the probability of blue nodes is equal to 1.

Lemma: A code is prefix-free if and only if each of its codewords is a leaf



source: http://reeves.ee.duke.edu/information_theory/lecture5-Lossless_Compression.pdf

Theorem: There exists a prefix-free code with length function $\ell(x)$ if only if $\ell(x)$ satisfies the Kraft Inequality

$$\ell(x) \text{ is the length function of a prefix-free code } \Leftrightarrow \sum_x D^{-\ell(x)} \leq 1$$

$$\begin{aligned} A &= \frac{1}{2^3} \\ B = C &= \frac{1}{2^4} \\ D = E = F &= \frac{1}{2^2} \\ Sum &= \frac{1}{2^3} + \frac{2}{2^4} + \frac{3}{2^2} < 1 \end{aligned}$$

▼ Shannon Code

Consider the following binary Shannon code. The entropy is $H(X) \approx 2.2855$ (bits) and the expected length is $E[\ell(X)] = 3.5$.

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25					
2	0.25					
3	0.2					
4	0.15					
5	0.15					

1. The source alphabet be $\mathcal{X} = 1, 2, \cdots, m$, and the cumulative distribution function (cdf) of the source distribution is

$$F(x) = \sum_{k \le x} p(k)$$

Then we update the table:

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25				
2	0.25	0.5				
3	0.2	0.7				
4	0.15	0.85				
5	0.15	1				

2. Let $\overline{F}(x)$ be the midpoint of the interval $[F(x-1), F(x)), i.e.$

$$\overline{F}(x) = \frac{F(x-1) + F(x)}{2} = F(x-1) + \frac{p(x)}{2}$$

Then we update the table:

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25	0.125			
2	0.25	0.5	0.375			
3	0.2	0.7	0.6			
4	0.15	0.85	0.775			
5	0.15	1	0.925			

3. We transfer the decimal to binary, i.e. for $x = 0.125$

$$0.125 * 2 = \underline{0}.25 \rightarrow 0$$
$$0.25 * 2 = \underline{0}.5 \rightarrow 0$$
$$0.5 * 2 = \underline{1} \rightarrow 1$$

Then we update the table as follows:

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25	0.125	0.001		
2	0.25	0.5	0.375	0.011		
3	0.2	0.7	0.6	$0.1\overline{0011}$		
4	0.15	0.85	0.775	$0.110\overline{0011}$		
5	0.15	1	0.925	$0.111\overline{01100}$		

4. The length function $\ell(x)$ is given by:

$$\ell(x) = \left\lceil \log \left(\frac{1}{p(x)} \right) \right\rceil$$

Then we update the table as follows:

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25	0.125	0.001	3	
2	0.25	0.5	0.375	0.011	3	
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	
5	0.15	1	0.925	$0.111\overline{01100}$	4	

5. The codeword corresponds to the D-ary expansion of the real number $\overline{F}(x)$, truncated at the point where the codeword is unique

$$C(x) = \text{D-ary expansion of } \overline{F}(x) \text{ such that } |C(x) - \overline{F}(x)| < \frac{1}{2}p(x)$$

$$\bar{F}(x) = \overbrace{0.\underbrace{z_1 z_2 \cdots z_{\ell(x)}}_{C(x)} z_{\ell(x)+1} z_{\ell(x)+2} \cdots}^{D\text{-ary expansion}}$$

Then we update the table as follows:

x	$p(x)$	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ in binary	$\left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	$C(x)$
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.5	0.375	0.011	3	011
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	1001
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	1100
5	0.15	1	0925	$0.111\overline{01100}$	4	1110

6. Here the expected length of the Shannon code obeys:

$$\mathbb{E}[\ell(X)] < \frac{H(X)}{\log D} + 2$$

Huffman Code

Recall that the Kraft inequality is a:

- necessary condition for uniquely decodable
- sufficient condition for the existence of a prefix-free code

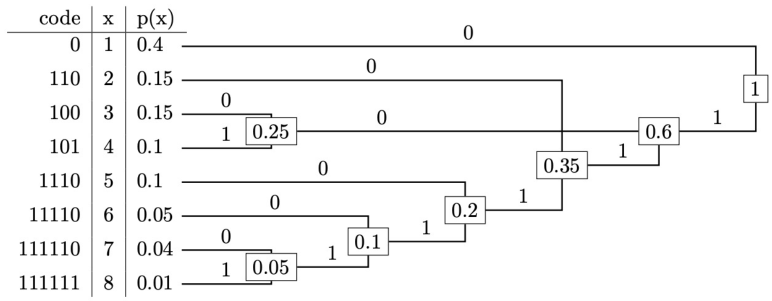
The search for the optimal code can be states as the following optimization problem. Given $p(x)$ find a length function $\ell(x)$ that minimizes the expected length and satisfies the Kraft inequality:

$$\min_{\ell(\cdot)} \sum_{x \in \mathcal{X}} p(x)\ell(x) \quad \text{s.t.} \quad \sum_{x \in \mathcal{X}} D^{-\ell(x)} \leq 1, \quad \ell(x) \text{ is an integer}$$

Construction of the Huffman Code:

1. Take the two least probable symbols. These are assigned the longest codewords which have equal length and differ only in the last digit.
2. Merge these two symbols into a new symbol with combined probability mass and repeat.

Theorem: Huffman's algorithm produces an optimal code tree



Lemma 1: In an optimal code, shorter codewords are assigned large probabilities, i.e.

$$p_i > p_j \Rightarrow \ell_i < \ell_j$$

Lemma 2: There exists an optimal code for which the codewords assigned to the smallest probabilities are siblings (i.e., they have the same length and differ only in the last symbol).