Entropy

As we have seen, the codeword lengths ℓ_s, $s \in \Sigma$, of a complete prefix code for an alphabet Σ satisfy $\sum_{s \in \Sigma} 2^{-\ell_s} = 1$. Thus the prefix code defines a probability distribution over Σ:

$$P(s) = 2^{-\ell_s} \text{ for all } s \in \Sigma.$$

Conversely, we can convert probabilities into codeword lengths:

$$\ell_s = -\log P(s) \text{ for all } s \in \Sigma.$$

For an arbitrary distribution, these codeword lengths are generally not integral and do not represent an actual prefix code. However, we can use these lengths to compute the average code length. The resulting quantity is called the entropy of the distribution.

Definition 1.24: Let P be a probability distribution over an alphabet Σ. The entropy of P is

$$H(P) = -\sum_{s \in \Sigma} P(s) \log P(s).$$

The quantity $-\log P(s)$ is called the self-information of the symbol s.

The concept of (information theoretic) entropy was introduced in 1948 by Claude Shannon in his paper "A Mathematical Theory of Communication" that established the discipline of information theory. From that paper is also the following result.

Theorem 1.25: (Noiseless Coding Theorem) Let C be an optimal prefix code for an alphabet Σ with the probability distribution P. Then

$$H(P) \leq \sum_{s \in \Sigma} P(s)|C(s)| < H(P) + 1.$$

The same paper contains the Noisy-channel Coding Theorem that deals with coding in the presence of error.

The entropy is an absolute lower bound on compressibility in the average sense. Because of integral code lengths, a prefix codes may not quite match the lower bound, but the upper bound shows that they can get fairly close. We will later see how to get even closer, but it is never possible to get below entropy.

Example 1.26: Continuing example 1.7, we get

symbol	a	e	i	o	u	y
probability	0.2	0.3	0.1	0.2	0.1	0.1
self-information	2.32	1.74	3.32	2.32	3.32	3.32
Huffman codeword length	2	2	3	2	4	4

$H(P) \approx 2.45$ while the average Huffman code length is 2.5.

One can prove tighter upper bounds on the average Huffman code length in terms of the maximum probability p_{max} of any symbol: $H(P) + p_{\text{max}} + 0.086$ when $p_{\text{max}} < 0.5$ and $H(P) + p_{\text{max}}$ when $p_{\text{max}} \geq 0.5$.

The case when p_{max} is close to 1, represents the worst case for prefix coding, because then the self-information $-\log p_{\text{max}}$ is much less than 1, but a codeword can never be shorter than 1.

Example 1.27:

symbol	a	b
probability	0.99	0.01
self-information	0.014	6.64
Huffman codeword length	1	1

$H(P) \approx 0.081$ while the average Huffman code length is 1. The Huffman code is more than 10 times longer than the entropy.

We will next prove the Noiseless Coding Theorem. The proof uses the following result.

Lemma 1.28: (Gibbs' inequality) For any two probability distributions P and Q over Σ

$$H(P) = -\sum_{s\in\Sigma} P(s) \log P(s) \le -\sum_{s\in\Sigma} P(s) \log Q(s).$$

with equality if only if $P = Q$.

Proof. Since $\ln x \le x - 1$ for all $x > 0$,

$$\sum_{s\in\Sigma} P(s) \ln(Q(s)/P(s)) \le \sum_{s\in\Sigma} P(s)(Q(s)/P(s) - 1) = \sum_{s\in\Sigma} Q(s) - \sum_{s\in\Sigma} P(s) = 0.$$

Dividing both sides by $\ln 2$, we can change ln into log. Then

$$\sum_{s\in\Sigma} P(s) \log Q(s) - \sum_{s\in\Sigma} P(s) \log P(s) = \sum_{s\in\Sigma} P(s) \log(Q(s)/P(s)) \le 0.$$

Rearrenging the terms, we get the inequality in the lemma.

Since $\ln x = x - 1$ if and only if $x = 1$, all the inequalities above are equalities if and only if $P = Q$.

$\square$

Proof of Theorem 1.24 (Noiseless Coding Theorem). Let us prove the upper bound first. For all $s \in \Sigma$, let $\ell_s = \lceil -\log P(s) \rceil$. Then

$$\sum_{s\in\Sigma} 2^{-\ell_s} \leq \sum_{s\in\Sigma} 2^{\log P(s)} = \sum_{s\in\Sigma} P(s) = 1$$

and by Kraft's inequality, there exists a prefix code with codeword length ℓ_s for each $s \in \Sigma$ (known as Shannon code). The average code length of this code is

$$\sum_{s\in\Sigma} P(s)\ell_s = \sum_{s\in\Sigma} P(s)\lceil -\log P(s) \rceil < \sum_{s\in\Sigma} P(s)(-\log P(s) + 1) = H(P) + 1.$$

Now we prove the lower bound. Let C be an optimal prefix code with codeword length ℓ_s for each $s \in \Sigma$. By Corollary 1.15, we can assume that C is complete, i.e., $\sum_{s\in\Sigma} 2^{-\ell_s} = 1$. Define a distribution Q by setting $Q(s) = 2^{-\ell_s}$. By Gibbs' inequality, the average code length of C satisfies

$$\sum_{s\in\Sigma} P(s)\ell_s = \sum_{s\in\Sigma} P(s)(-\log(2^{-\ell_s})) = -\sum_{s\in\Sigma} P(s)\log(Q(s)) \geq H(P).$$

$\square$

Arithmetic coding

Arithmetic coding can be seen as prefix coding with fractional code lengths, or as prefix coding of strings instead of symbols. Arithmetic coding can get below the average code length of Huffman coding and arbitrarily close to the entropy.

Let $\Sigma = \{s_1, s_2, \ldots, s_\sigma\}$ be an alphabet with an ordering $s_1 < s_2 < \cdots < s_\sigma$. Let P be a probability distribution over Σ and define the cumulative distribution F as

$$F(s_i) = \sum_{j=1}^{i} P(s_j) \text{ for all } i \in [1, \ldots, \sigma]$$

We associate a symbol s_i with the interval $[F(s_i) - P(s_i), F(s_i))$.

Example 1.29:

symbol	a	b	c
probability	0.2	0.5	0.3
interval	$[0, 0.2)$	$[0.2, 0.7)$	$[0.7, 1.0)$

We can extend the interval representation from symbols to strings. For any $n \geq 1$, define a probability distribution P_n and a cumulative distribution F_n over Σ^n, the set of strings of length n, as

$$P_n(X) = \prod_{i=1}^{n} P(x_i)$$
$$F_n(X) = \sum_{Y \in \Sigma^n, Y \leq X} P(Y)$$

for any $X = x_1 x_2 \ldots x_n \in \Sigma^n$, where $Y \leq X$ uses the lexicographical ordering of strings. The interval for X is $[F_n(X) - P_n(X), F_n(X))$.

The interval can be computed incrementally.

Input: string $X = x_1 x_2 \ldots x_n \in \Sigma^n$ and distributions P and F over Σ.
Output: interval $[l, r)$ for X.

```
(1)  $[l, r) \leftarrow [0.0, 1.0)$
(2)  for $i = 1$ to $n$ do
(3)      $r' \leftarrow F(x_i)$; $l' \leftarrow r' - P(x_i)$        // $[l', r')$ is interval for $x_i$
(4)      $p \leftarrow r - l$; $l \leftarrow l + p \cdot l'$; $r \leftarrow l + p \cdot r'$
(5)  return $[l, r)$
```

Example 1.30: Continuing Example 1.29, the interval for the string `cab` is computed as follows:

symbol	`a`	`b`	`c`
probability	0.2	0.5	0.3
interval	$[0, 0.2)$	$[0.2, 0.7)$	$[0.7, 1.0)$

string	`ca`	`cb`	`cc`
probability	0.06	0.15	0.09
interval	$[0.7, 0.76)$	$[0.76, 0.91)$	$[0.91, 1.0)$

string	`caa`	`cab`	`cac`
probability	0.012	0.03	0.018
interval	$[0.7, 0.712)$	$[0.712, 0.742)$	$[0.742, 0.76)$

We will similarly associate intervals to sequences over the code alphabet $\{0,1\}$ using the uniform distribution. The interval for a binary string $B \in \{0,1\}^\ell$ is

$$\left[\frac{\mathrm{val}(B)}{2^\ell}, \frac{\mathrm{val}(B)+1}{2^\ell}\right)$$

where $\mathrm{val}(B)$ is the value of B as a binary number.

- This is the same relation between binary strings and dyadic intervals that we already saw in the proof of Kraft's inequality. Recall that a set of binary strings forms a prefix code if and only if their intervals do not overlap each other.

- The interval for B contains exactly the numbers whose representation in binary begins with B (after "."). In fact, in the literature, arithmetic coding is usually described using binary fractions instead of intervals.

Example 1.31: For $B = 101110$, $\mathrm{val}(B) = 46$ and thus the interval is

$$\left[\frac{46}{64}, \frac{47}{64}\right) = [.71875, .734375) = [.101110, .101111).$$

The missing step of arithmetic encoding is to connect source alphabet intervals with code alphabet intervals. There are a few different ways to do this. Our solution is to map a source interval $[l, r)$ into the largest dyadic interval that is completely contained in $[l, r)$.

Now we have the full encoding procedure:

$$\text{source string} \mapsto \text{source interval} \mapsto \text{code interval} \mapsto \text{code string}$$

Example 1.32: The string `cab` is encoded as 101110:

$$\texttt{cab} \mapsto [.712, .742) \mapsto [.71875, .734375) = [.101110, .101111) \mapsto 101110$$

The mapping from source intervals to code intervals ensures that the code intervals do not overlap. Therefore, we have a prefix code for Σ^n.

The prefix code is usually not optimal or even complete, because there are gaps between the code intervals. However, the length of the code interval for a source interval of length p is always more than $p/4$. Thus the code length for a string X is less than $-\log P_n(X) + 2$, and the average code length is less than $H(P_n) + 2$.