Note 10. Information theory II

Alex Fu

Fall 2022

6 Source coding

Let us now return to the two central questions of communication. Shannon's key insight was that the two steps of *source coding* and *channel coding* can be designed separately, and together still achieve the optimal rate of transfer of information.

6.1 Lossless compression

Definition 1 (Source coding scheme; message; length).

A **source coding scheme** is a function $S \to \{0,1\}^*$ from the source alphabet to all possible binary strings of finite length.

A **message** is a sequence of symbols $(x_1, \ldots, x_n)$. The probability of observing a message is

$$p_{X^{(n)}}(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_X(x_i),$$

and the **length** of a message $\text{len}(x_1, \ldots, x_n)$ is the number of bits in its encoded sequence.

The goal of **lossless compression** is to minimize $\mathbb{E}(\text{len}(x_1, \ldots, x_n))$, with the condition that given an encoded or compressed sequence of bits, the original message can still be recovered.

The key result: the **entropy** of X is a fundamental asymptotic lower bound on the average number of bits per symbols to which *any* source coding scheme can compress a message. For a sense of where $H(X)$ comes from, by the LLN, a "typical message" has probability

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p_i^{np_i} = \prod_{i=1}^{n} 2^{np_i \log_2(p_i)} = 2^{\sum_{i=1}^{n} np_i \log_2(p_i)} = 2^{-nH(X)}.$$

6.2 Source coding theorem

The proofs in the following derivation will be left to you as homework.

Proposition 1 (Asymptotic equipartition property).

Asymptotically, the probability mass of the set of all possible sequences is concentrated on a subset *partitioned* into $2^{nH(X)}$ sequences, each with *equal* probability $2^{-nH(X)}$:

$$\lim_{n\to\infty} -\frac{1}{n}\log_2 p(x_1,\ldots,x_n) \overset{\text{a.s.}}{=} H(X).$$

Definition 2 (Typical set).

Let $\varepsilon > 0$. The **typical set** of sequences of length n is

$$A_\varepsilon^{(n)} := \left\{(x_1,\ldots,x_n) \in S^n : 2^{-n(H(X)+\varepsilon)} \le p(x_1,\ldots,x_n) \le 2^{-n(H(X)-\varepsilon)}\right\}.$$

In what sense is the typical set $A_\varepsilon^{(n)}$ typical?

Proposition 2 (Asymptotic typicality).

For all sufficiently large n,

$$\mathbb{P}\big(A_\varepsilon^{(n)}\big) = \mathbb{P}\big((x_1,\ldots,x_n) \in A_\varepsilon^{(n)}\big) > 1-\varepsilon.$$

Moreover, the typical set can be *exponentially smaller* in size than the set of all sequences S^n:

Proposition 3 (Size of the typical set).

For all sufficiently large n,

$$(1-\varepsilon)\cdot 2^{n(H(X)-\varepsilon)} \le \big|A_\varepsilon^{(n)}\big| \le 2^{n(H(X)+\varepsilon)}.$$

Furthermore, the typical set cannot be any smaller:

Proposition 4 (Minimality of the typical set).

Let $\delta > 0$, and let $B_\delta^{(n)}$ be of size at most $2^{n(H(X)-\delta)}$. Then, for all sufficiently large n,

$$\mathbb{P}\Big(B_\delta^{(n)}\Big) = \mathbb{P}\Big((x_1,\ldots,x_n) \in B_\delta^{(n)}\Big) < \delta.$$

There may appear to be a slight contradiction between Proposition 3 and 4; how does the order of choosing δ and ε resolve this?

The final main result, due to Shannon, allows us to interpret entropy as the expected information content of a distribution — information is lost if compression goes below the entropy.

Theorem 1 (Source coding theorem).

Any sequence of symbols can be losslessly compressed such that asymptotically, the expected number of bits per symbol is arbitrarily close to the entropy, and the probability of error is arbitrarily small. Conversely, no source coding can achieve an expected number of bits per symbol less than the entropy without almost sure loss.

Formally, let $\varepsilon > 0$, and let $(X_i)_{i=1}^n$ be symbols i.i.d. according to p_X. Then there exists a source coding scheme $S \to \{0,1\}^*$ such that

$$\lim_{n\to\infty} \mathbb{E}\left(\frac{1}{n}\,\mathsf{len}(X_1,\dots,X_n)\right) = \lim_{n\to\infty} \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\mathsf{len}(X_i)\right) \le H(X) + \varepsilon,$$

and the original message $(X_1,\dots,X_n)$ can be recovered with probability $1-\varepsilon$.

For an asymptotically error-free encoding (by Proposition 2), we can use $\log_2 |A_\varepsilon^{(n)}|$ bits to encode the typical messages, and ignore all other messages. Or, we can add 1 bit to indicate if a sequence is typical, and encode the atypical messages using at most $n\log_2|S|$ bits.

The latter always error-free encoding requires at most

$$(1-\varepsilon)\cdot\left(1+\left\lceil\log_2\left|A_\varepsilon^{(n)}\right|\right\rceil\right) + \varepsilon\cdot\left(1+n\left\lceil\log_2|S|\right\rceil\right)$$

bits in expectation to encode a message. We can show that this scheme can also get arbitrarily close to entropy (the *Shannon limit*) using an $\varepsilon/2$ trick.

6.3 Huffman coding

On this note, let us consider actual practical implementations of source coding, one of the most common of which is *Huffman coding*. In the following discussion, an encoding scheme is simply a *code*, and a *codeword* is the encoded bitstring of some symbol in S.

Definition 3 (Prefix code; unique decodability).

A **prefix code** is *prefix-free* — no codeword is a prefix of another codeword. Equivalently, a code is a prefix code if every codeword can be described by a path from the root to a leaf in some fixed binary tree.

The prefix-free property implies that any sequence of bits can be **uniquely decoded**, though not necessarily the converse.

Example 1 (Prefix and non-prefix codes).

The set of codewords $\{0, 10, 110, 111\}$ is a prefix code. $\{0, 10, 010\}$ is not prefix-free, and 010 is not uniquely decodable. However, there also exist non-prefix codes that are nonetheless uniquely decodable, such as $\{0, 11, 010\}$.

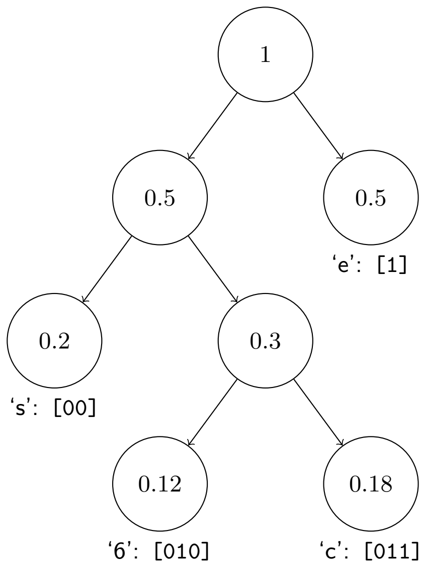
Definition 4 (Huffman coding).

Huffman coding is a greedy algorithm that builds an optimal prefix code, which turns out to be optimal among all uniquely decodable codes that assign codewords symbol-by-symbol.

```
def huffman(p_1, ..., p_n → Q):
    input: the probabilities of each symbol
    output: the encoding binary tree
    Q ← priority queue of nodes sorted by p_i
    for i = 1 to n:
        add leaf node i to Q
    while size of Q is not 1:
        i ← pop min from Q
        j ← pop min from Q
        create new node x
        x.left, x.right, p_x ← i, j, p_i + p_j
        add x to Q with priority p_x
```

The algorithm constructs bottom-up a binary tree whose nodes are *metacharacters*, sets of one or more symbols. Each symbol is a leaf, with its codeword given by the path to it starting from the root, where conventionally the left branch is 0 and the right branch is 1.

The following example of a Huffman tree shows that ties can be broken arbitrarily. What property of the binary tree guarantees that the resulting code is prefix-free?



The idea behind Huffman coding is quite similar to deterministic *binary search*, in which a search space of k objects is repeatedly halved. Every object uniquely determines $\lceil \log_2(k) \rceil$ yes-no answers to the questions of "left or right half" in each iteration, which gives the object's bitstring.

However, with Huffman coding, the objects or symbols are not equally weighted in probability, so more probable symbols should have shorter bitstrings to minimize the expected length. Thus we split the "probability search space" of total mass 1 approximately in half with each question.

Example 2 (Optimality of Huffman coding).

If every p_i is a power of $\frac{1}{2}$, then the expected number of bits per symbol under Huffman coding is precisely the entropy. The *depth* of a leaf — the number of nodes to it from the root, or the length of its codeword — is precisely ℓ_i if $p_i = 2^{-\ell_i}$, so the expected length is

$$\sum_{i=1}^{n} p_i \ell_i = \sum_{i=1}^{n} p_i \log_2\left(\frac{1}{p_i}\right) = H(X).$$

In general, there are more optimal codes such as *arithmetic coding* that encode entire messages rather than individual symbols, but these are out of our scope for now.

7 Channel coding

Our aim in this section is the transmission of encoded information over a noisy channel, so our setup will be slightly different from source coding.

Definition 5 (Channel).

A **channel** is a conditional probability distribution $p_{Y|X}(y \mid x)$, which is used n times for an encoded message of n bits. $X \in \{0, 1\}$ is the input bit, and Y is the output bit.

Example 3 (Binary symmetric channel; binary erasure channel).

The **binary symmetric channel** (BSC) *errors* (sends the opposite bit) with probability p, and the **binary erasure channel** (BEC) *erases* or *corrupts* an input bit with probability p, which is represented as sending the special symbol $?$ or ε.

$$\begin{tikzcd}
0 \arrow[r, "1-p"] \arrow[dr, "p"] & 0 \\
1 \arrow[r, "1-p"'] \arrow[ur, "p"'] & 1
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
0 \arrow[r, "1-p"] \arrow[dr, "p"] & 0 \\
 & ? \\
1 \arrow[r, "1-p"'] \arrow[ur, "p"] & 1
\end{tikzcd}$$

Definition 6 (Capacity).

The **capacity** of a channel described by $p_{Y|X}$ is

$$C := \max_{p_X} I(X;Y).$$

In the ideal case $Y = X$, no information content of X is lost during transmission: $C = I(X;Y) = H(X)$. Perhaps surprisingly, $Y = 1 - X$ is also ideal, as we can imagine always flipping the output bit to recover X. In general, the capacity falls between 0 and $H(X)$.

Theorem 2 (Capacity of the BSC).

The capacity of the binary symmetric channel with error probability p is $1 - H_b(p)$.

Proof. We can find the capacity directly:

$$\begin{aligned}
C &= \max_{p_X} H(Y) - H(Y \mid X) \\
&= \max_{p_X} H(Y) - \sum_{x \in \{0,1\}} p_X(x) \cdot H(Y \mid X = x) \\
&= \max_{p_X} H(Y) - \sum_{x \in \{0,1\}} p_X(x) \cdot H_b(p) \\
&= \max_{p_X} H(Y) - H_b(p) \\
&= 1 - H_b(p),
\end{aligned}$$

where $H(Y) = 1$ is maximized by uniform p_X. □

Definition 7 (Encoding function; decoding function; probability of error).

Suppose we want to send a message of length ℓ. A lossless source coding scheme $S \to \{0,1\}^*$ gives an **encoding function** $f_n \colon S^\ell \to \{0,1\}^n$ and a **decoding function** $g_n \colon \{0,1,?\}^n \to S^\ell$. If the message M is encoded as the input bitstring $f_n(M) = (x_1, \ldots, x_n)$, and the output of the channel is $(y_1, \ldots, y_n)$, then the **probability of error** is

$$\mathbb{P}_{\text{error}}(n) := \mathbb{P}(g_n(y_1, \ldots, y_n) \neq (x_1, \ldots, x_n) \mid f_n(M) = (x_1, \ldots, x_n)).$$

We naturally want the channel to be *asymptotically error-free*, so that its $\mathbb{P}_{\text{error}}(n)$ tends to 0 as $n \to \infty$, just as $\mathbb{P}((x_1, \ldots, x_n) \notin A_\varepsilon^{(n)}) \to 0$ for source coding. We next find that capacity is an optimal upper bound on the *rate* of transmission, just as entropy is an optimal lower bound on the expected length of compression, both asymptotically error-free.

Definition 8 (Rate).

The **rate** of a channel is the ratio

$$R := \frac{\ell}{n} = \frac{H(M)}{n}.$$

That is, the number of bits of information about the source message per symbol received.

Definition 9 (Achievability).

Channel rate R is **achievable** if for every $n \in \mathbb{Z}^+$, there exists an encoding function f_n and a decoding function g_n (where messages have length $\ell(n) := \lceil nR \rceil$) such that the channel is asymptotically error-free.

The following central result of channel coding justifies the equivalent definition that the capacity of a channel is its *largest achievable rate*, though the proof is out of our scope.

Theorem 3 (Channel coding theorem).

Any rate below the channel capacity is achievable. For any channel with capacity C and any $R < C$, there exists a communication scheme with rate R that is asymptotically error-free.

Conversely, no rate greater than the capacity is achievable without a non-negligible probability of error. Furthermore, any transmission rate up to

$$R(\varepsilon) := \frac{C}{1 - H_b(\varepsilon)}$$

is achievable if and only if the probability of error is at most ε.

Theorem 4 (Capacity of the BEC).

The capacity of the binary erasure channel with error probability p is $1 - p$.

Proof. We show that no rate above $1-p$ is achievable by an **oracle argument**. Even supposing perfect feedback from the receiver, the transmitter can only resend the erased input bits, of which p will be erased again. Thus the reliable rate of communication is $1 - p$ bits per channel use.

Now, let $\varepsilon > 0$, and let us show that any $R = 1 - p - \varepsilon$ is achievable. We generate a table with $2^{\ell(n)}$ rows and n columns, where each entry is i.i.d. Bernoulli($\frac{1}{2}$). The **codebook** thus associates each of the $2^{\ell(n)}$ possible messages to a codeword of length n.

By the Strong Law of Large Numbers, we can consider the channel to erase p of the bits in any codeword, and we suppose without loss of generality that the first $\lfloor n(1-p) \rfloor$ bits are not erased. The receiver then truncates their codebook to the first $\lfloor n(1-p) \rfloor$ columns, and declares an error if there is no *unique* match between the received bits and a truncated codeword.

Thus, the probability of error is the probability that at least two entries in the truncated codebook match the received bits. (By construction, no truncated codewords match the received bits with negligible probability.) If γ_i are the truncated codewords, then by the union bound,

$$\mathbb{P}_{\text{error}}(n) = \mathbb{P}\left(\bigcup_{i=2}^{2^{\ell(n)}} \{\gamma_1 = \gamma_i\}\right) \leq 2^{\ell(n)} \cdot 2^{-\lfloor n(1-p) \rfloor} \simeq 2^{-n(1-p-R)} \to 0.$$

□

In practice, however, exhaustive search over a massive codebook is unusable, even if the communication scheme does achieve channel capacity. Thus, implementable and fast codes such as *error-correcting codes* are needed to approach the optimal rate of transmission.

■