Source Representation: Lossless Compression 5

5.1 Rate-distortion Function at Zero Expected Distortion 65
5.2 Variable-length Coding for Exactly Lossless Compression 67
5.3 Unique Decodability and Kraft Inequality . . 69
5.4 Shannon-Fano Code and Huffman Code 73

In the previous lecture, we have characterized the fundamental tradeoff between rate and distortion for source representation, as given by the rate-distortion function. In many applications, it is desirable or even mandatory to have perfect reproduction of the source message, i.e., lossless compression. The rate-distortion function specialized to zero expected distortion corresponds to asymptotically lossless compression. Furthermore, if exactly lossless compression is required, variable length coding becomes necessary. Interestingly, both asymptotically lossless compression and exactly lossless compression share the same ultimate fundamental performance limit, given by the entropy of source. These topics are treated in this lecture.

5.1 Rate-distortion Function at Zero Expected Distortion

Recall that for the lossy case treated in Lecture 4, the rate-distortion function of a DMS S under distortion measure d is characterized by Theorem 4.1, Shannon's fundamental theorem for source coding, as

$$R(D) \quad = \quad \min_{P_{\hat{S}|S}} I(S;\hat{S}) \tag{5.1}$$

$$\text{s.t.} \qquad \mathbf{E}[d(S,\hat{S})] \leq D. \tag{5.2}$$

Now, suppose that $\mathcal{S} = \hat{\mathcal{S}}$, and that the distortion measure is Hamming. Let us focus on $R(0)$.

When $D = 0$, since the distortion measure is Hamming, the constraint $\mathbf{E}[d(S,\hat{S})] \leq D = 0$ becomes $P(S \neq \hat{S}) = 0$. So we have $\hat{S} = S$ with probability one in solving $R(0)$. This immediately leads to

$$I(S;\hat{S}) = I(S;S) = H(S). \tag{5.3}$$

We hence conclude that $R(0) = H(S)$, the entropy of the DMS S.

Remark 5.1 Note that under the problem formulation in Section I of Lecture 4, $D = 0$ means

$$\lim_{n\to\infty} \mathbf{E}[d(\underline{S},\underline{\hat{S}})] = 0; \tag{5.4}$$

that is, for any $\epsilon > 0$, for all sufficiently large n,

$$\frac{1}{n}\sum_{i=1}^{n} P(S_i \neq \hat{S}_i) \leq \epsilon. \tag{5.5}$$

This only implies that the fraction of successfully reproduced source symbols approaches 100% asymptotically as the source message length grows without bound. But it is insufficient for us to assert that exactly lossless compression, i.e.,

$$P(\underline{S} = \underline{\hat{S}}) = 1, \tag{5.6}$$

is possible, under the lossy source representation problem formulation. It is important to keep this subtlety in mind.

We can get some further insight by revisiting the proof of the achievability part of Theorem 4.1. Therein, the elements of the codebook $\mathsf{\mathbf{C}}$ are i.i.d. random variables obeying probability distribution $P_{\hat{S}}$. Now, since $\hat{S} = S$, we have $P_{\hat{S}} = P_S$. So $\mathsf{\mathbf{C}}$ simply consists of i.i.d. samples of S.

Let us check the encoding procedure at $D = 0$. Since the distortion measure is Hamming and $S = \hat{S}$, in (4.39), we have

$$\begin{aligned} d(\underline{s}, \mathsf{C}(w)) &= \frac{1}{n}\sum_{i=1}^{n} d(s_i, \mathsf{C}_i(w)) \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{s_i \neq \mathsf{C}_i(w)\}}, \end{aligned} \tag{5.7}$$

namely, the fraction of positions in which $\underline{s}$ and $\mathsf{C}(w)$ disagree, and the criterion (4.39) requires that this fraction is small, specifically, no greater than ϵ.

On the other hand, in (4.40), since $\hat{S} = S$, the information density between $\underline{s}$ and $\mathsf{C}(w)$ is

$$\begin{aligned} i(\underline{s}; \mathsf{C}(w)) &= \frac{1}{n}\sum_{i=1}^{n} \log \frac{P_{S,S}(s_i, \mathsf{C}_i(w))}{P_S(s_i)P_S(\mathsf{C}_i(w))} \\ &= \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{P_S(s_i)}, \end{aligned} \tag{5.8}$$

if $\underline{s} = \mathsf{C}(w)$, and $-\infty$ otherwise. This is because even a single position in which $s_i \neq \mathsf{C}_i(w)$ will lead to $P_{S,S}(s_i, \mathsf{C}_i(w)) = 0$. So the criterion (4.40) requires that $\underline{s} = \mathsf{C}(w)$ holds, and furthermore $\frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{P_S(s_i)}$ lies within $[H(S) - \epsilon, H(S) + \epsilon]$.

Here, it is clear that (4.40) dominates (4.39). So given a source message $\underline{s}$, the encoding procedure searches among the codebook C for a codeword that is exactly $\underline{s}$, and declares an encoding failure if $\underline{s}$ is not found in C. The proof in Section 4.4 ensures that, by choosing any rate $R > R(0) = H(S)$, for any $\epsilon > 0$, as $n \longrightarrow \infty$, the probability that encoding failures occur tends to zero. How to understand this result intuitively? Of course, it is impossible to use a codebook of rate $R < \log|\mathcal{S}|$ to exhaustively contain all possible source messages, but by generating the codebook at random according to P_S, it is ensured that those source messages not contained in the codebook occur with arbitrarily small probability, as the source message length grows sufficiently large.

5.2 Variable-length Coding for Exactly Lossless Compression

As remarked in the previous section, $R(0) = H(S)$ implies that asymptotically lossless compression is possible for any rate greater than $H(S)$. But if exactly lossless compression is required, i.e., $\hat{\underline{S}} = \underline{S}$ to hold with probability one, we need a different approach.

The basic idea is to exploit the fact that the probability distribution of S is generally not uniform. Therefore, variable-length coding comes into play. Recall the problem formulation in Section 4.1, where a source message $\underline{S}$ is encoded by an encoding mapping as an index W, and W is then decoded by a decoding mapping into a reproduced message $\hat{\underline{S}}$. With variable-length coding, we consider W as a binary or q-ary string, and allow its length to vary for different source messages. Intuitively, by allocating a shorter (resp. longer) index string to a source message with higher (resp. lower) probability, the overall efficiency of coding, measured by the expected index string length, can be improved.

Consider a DMS S, and without loss of generality, arrange its alphabet $\mathcal{S} = \{a_1, a_2, \ldots\}$ so that the probabilities are non-increasing, i.e., $P_S(a_1) \geq P_S(a_2) \geq \ldots$. We encode S into an index $W = f(S)$, which, when represented as a binary string, is drawn from

$$\mathcal{W}^* = \{\emptyset, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}, \tag{5.9}$$

the set of all finite-length binary strings, including the empty string $\emptyset$. In this section we only consider the binary case, and the extension to q-ary cases is left as an exercise.

In order to minimize the expected index length, arguably the coding rule is to always assign a shorter index string to a more frequently occurring source message. That is, $f(a_1) = \emptyset$, $f(a_2) = 0$, $f(a_3) = 1$, $f(a_4) = 00$, $f(a_5) = 01, \ldots$ We can readily verify that, for source message a_i, its index string has length $\ell(a_i) = \lfloor \log_2 i \rfloor$.

Let us analyze the expected index string length, $\bar{\ell} = \sum_{s \in \mathcal{S}} P_S(s)\ell(s)$.

Regarding the upper bound of $\bar{\ell}$, note that for a_i, $P_S(a_i) \le 1/i$. So we have

$$\ell(a_i) = \lfloor \log_2 i \rfloor \le \log_2 i \le -\log_2 P_S(a_i), \tag{5.10}$$

and

$$\begin{aligned} \bar{\ell} &= \sum_{s \in \mathcal{S}} P_S(s)\ell(s) \\ &\le -\sum_{s \in \mathcal{S}} P_S(s) \log_2 P_S(s) = H(S) \text{ in bits.} \end{aligned} \tag{5.11}$$

Regarding the lower bound of $\bar{\ell}$, we proceed as

$$\begin{aligned} H(S) &\overset{(a)}{=} H(S) + H(\ell(S)|S) \\ &\overset{(b)}{=} H(S, \ell(S)) \\ &\overset{(c)}{=} H(S|\ell(S)) + H(\ell(S)), \end{aligned} \tag{5.12}$$

where (a) is due to Corollary 3.6 because $\ell(S)$ is deterministic conditioned upon S, and (b) and (c) are both from the chain rule of entropy, i.e., Theorem 3.1.

For $H(S|\ell(S))$, note that for each $\ell(S) = \ell$, there are at most 2^ℓ different possibilities of S, because otherwise at least two source messages will be assigned to the same index string and the coding will not be exactly lossless. So

$$\begin{aligned} H(S|\ell(S)) &= \sum_{\ell=0}^{\infty} P(\ell(S) = \ell) H(S|\ell(S) = \ell) \\ &\le \sum_{\ell=0}^{\infty} P(\ell(S) = \ell)\ell \\ &= \mathbf{E}[\ell(S)] = \bar{\ell} \text{ in bits.} \end{aligned} \tag{5.13}$$

For $H(\ell(S))$, we have

$$\begin{aligned} H(\ell(S)) &\overset{(a)}{\le} (\bar{\ell} + 1)\log_2(\bar{\ell} + 1) - \bar{\ell} \log_2 \bar{\ell} \\ &\overset{(b)}{<} \log_2[e(\bar{\ell} + 1)] \\ &\overset{(c)}{\le} \log_2[e(H(S) + 1)] \text{ in bits,} \end{aligned} \tag{5.14}$$

where (a) is by using the property that geometric distribution maximizes entropy under a given mean, i.e., Corollary 3.2, (b) is from $(x + 1)\ln(x + 1) - x \ln x < \ln(x + 1) + 1$ for $x > 0$, and (c) is from the upper bound of $\bar{\ell}$ we just obtained in (5.11).

Returning to (5.12), we have

$$\begin{aligned} H(S) &= H(S|\ell(S)) + H(\ell(S)) \\ &< \bar{\ell} + \log_2[e(H(S)+1)]; \end{aligned} \tag{5.15}$$

that is,

$$\bar{\ell} > H(S) - \log_2[e(H(S)+1)]. \tag{5.16}$$

In summary, we have the following estimate of $\bar{\ell}$:

$$H(S) - \log_2[e(H(S)+1)] < \bar{\ell} \leq H(S) \text{ in bits.} \tag{5.17}$$

Now we can apply the result to a length-n source message $\underline{S}$, and correspondingly normalize the expected index string length by n, to estimate the expected index string length per source symbol, consistent with the notion of rate in the problem formulation in Section 4.1 of Lecture 4. Our upper bound (5.11) and lower bound (5.16) on $R = \bar{\ell}/n$ thus become

$$\begin{aligned} R &\leq \frac{H(\underline{S})}{n} = H(S), && (5.18) \\ R &> \frac{H(\underline{S})}{n} - \frac{\log_2[e(H(\underline{S})+1)]}{n} \\ &= H(S) - \frac{\log_2[e(nH(S)+1)]}{n}, && (5.19) \end{aligned}$$

in bits, respectively. As $n \to \infty$, we have $R \to H(S)$ (assuming $H(S) < \infty$). So we conclude that the fundamental performance limit of variable-length coding for exactly lossless compression is $H(S)$, which is the same as that for asymptotically lossless compression, i.e., $R(0)$, obtained in the previous section.

5.3 Unique Decodability and Kraft Inequality

The code described in the previous section is feasible and optimal when the size of the source message is prescribed. It may fail, however, if a source message with a size not prescribed in advance, is to be coded. Consider, for example, to perform exactly lossless compression for two different source messages $\underline{s} = [a_2, a_2, a_2, a_3]$ and $\underline{s}' = [a_4, a_5]$. According to the coding rule in the previous section, we find that both $\underline{s}$ and $\underline{s}'$ are encoded into the same index string 0001. So it is clear that there exists some ambiguity when applying the code to the "streaming" scenario where the source message itself has variable length, and therefore some further restrictions on feasible codes should be imposed.

Definition 5.1 For a DMS S, its q-ary index $W = f(S)$ is a finite-length string drawn from $\{0, 1, \ldots, q-1\}^* := \bigcup_{n=1}^{\infty}\{0, 1, \ldots, q-1\}^n$. The length of $W = f(S)$ is denoted by $\ell(S)$. The mapping $f : \mathcal{S} \mapsto \{0, 1, \ldots, q-1\}^*$ is called the base code for S.

Note that unlike in the previous section, here the empty string $\emptyset$ is excluded from the code.

Definition 5.2 Given a DMS S and its base code f, for any finite-length source message $\underline{s} = [s_1, s_2, \ldots, s_n]$, we define $f(\underline{s})$ as the symbol-by-symbol concatenation of base codes, i.e.,

$$f(\underline{s}) = [f(s_1), f(s_2), \ldots, f(s_n)], \tag{5.20}$$

for any integer $n \geq 1$ and any $\underline{s} \in \mathcal{S}^n$. This way, the mapping f is extended to $\mathcal{S}^*$, all finite-length source messages, and we can thus call it the code for S.

Definition 5.3 A code f for a DMS S is called non-singular if it is injective in the context of Definition 5.1, i.e., for any $s \neq s' \in \mathcal{S}$, $f(s) \neq f(s')$.

Remark 5.2 Although non-singularity seems to be natural in the lossless case, it is not entirely trivial, because for the general case of lossy source representation in Lecture 4, as illustrated in the achievability proof therein, the encoding cannot be non-singular in general.

Non-singularity, unfortunately, is insufficient when we extend the code f from $\mathcal{S}$ to $\mathcal{S}^*$. Accordingly, we need to extend the concept of non-singularity to $\mathcal{S}^*$ in light of Definition 5.2.

Definition 5.4 A code f for a DMS S is called uniquely decodable if it is injective in the context of Definition 5.2, i.e., for any two different finite-length source messages, $\underline{s}$ and $\underline{s}'$, $f(\underline{s}) \neq f(\underline{s}')$.

There exist algorithms (e.g., [23]) that check whether a given code is uniquely decodable, but a general uniquely decodable code may still be not very convenient to use. Next we focus on a special class of uniquely decodeable codes, i.e., those satisfying the so-called prefix-free property.

Definition 5.5 A code for a DMS S is called prefix-free if for any $s \neq s' \in \mathcal{S}$, $f(s)$ is not a prefix of $f(s')$.

For example, if a code consists of index strings $\{00, 01, 001\}$, then it is not prefix-free, because 00 is a prefix of 001; if we modify this code to $\{00, 01, 101\}$, then it becomes prefix-free.

The following lemma is immediate.

Lemma 5.1 If a code is prefix-free, then it is uniquely decodable.

A convenient feature of prefix-free codes is that, when we scan from the beginning to decode the concatenated string $f(\underline{s}) = [f(s_1), f(s_2), \ldots, f(s_n)]$, every time recognizing a string belonging to $\{f(s) : s \in \mathcal{S}\}$, we can immediately decode its corresponding source symbol. For this reason, a prefix-free code is also called an instantaneous code.

Example 5.1 For a uniquely decodable but not prefix-free code consisting of $\{0, 01, 11\}$, when decoding 01101 (encoded from $[a_1, a_3, a_2]$), we can be sure that the first 0 corresponds to a_1 only until scanning 011; but for a prefix-free code consisting of $\{0, 10, 11\}$, when decoding 01110 (also encoded from $[a_1, a_3, a_2]$), since there is no string 01 in the code, upon reading the first 0 we can immediately decode a_1.

Another convenient feature of prefix-free codes is that each prefix-free code for a DMS S with $|\mathcal{S}| < \infty$ can be visualized by a rooted q-ary tree, and each of its index strings corresponds to a unique leaf of the tree. There may be some unused leaves, but no non-leaf node can be an index string due to the prefix-free property. For $s \in \mathcal{S}$, the length of $f(s)$, $\ell(s)$, is exactly the depth of the leaf corresponding to $f(s)$ in the tree. Figure 5.1 gives an example of rooted binary tree visualization of a binary prefix-free code.

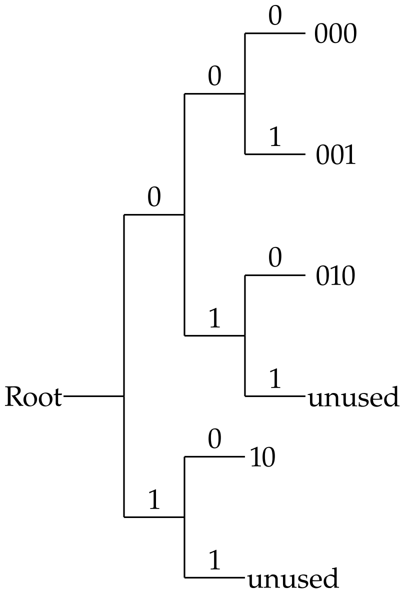

Figure 5.1: A rooted tree visualization of a binary prefix-free code.

It is clear that there can be many different prefix-free codes for a DMS S. Is there any fundamental restriction on the lengths their index strings? The answer is given by the following theorem.

Theorem 5.1 For a DMS S with $|\mathcal{S}| < \infty$, any of its prefix-free codes must satisfy the Kraft inequality

$$\sum_{s \in \mathcal{S}} q^{-\ell(s)} \leq 1; \tag{5.21}$$

conversely, for any $\{\ell(s) : s \in \mathcal{S}\}$ satisfying (5.21), there exists a prefix-free code with these index string lengths.

Proof: Recall that a prefix-free code can be visualized by a rooted q-ary tree. The deepest leaf in the tree corresponds to the longest index string, and let us denote this depth by $\ell_{\max}$, which is finite by the assumption of $|\mathcal{S}| < \infty$. For each $s \in \mathcal{S}$, its index string $f(s)$ is at depth $\ell(s)$ in the tree. If we grow the tree from the leaf $f(s)$ until reaching the maximum depth $\ell_{\max}$, then this procedure will produce $q^{\ell_{\max}-\ell(s)}$ leaves at depth $\ell_{\max}$. Due to the prefix-free property, all the leaves extended by any two different index strings do not overlap. Therefore, we have

$$\sum_{s \in \mathcal{S}} q^{\ell_{\max}-\ell(s)} \leq q^{\ell_{\max}}, \tag{5.22}$$

where the right hand side term $q^{\ell_{\max}}$ is the total number of leaves at depth $\ell_{\max}$ when we fully grow the tree. Dividing both sides by $q^{\ell_{\max}}$ leads to the Kraft inequality (5.21).

For the converse part, consider any $\{\ell(s) : s \in \mathcal{S}\}$ satisfying (5.21), and rearrange the elements of $\mathcal{S}$ so that $\ell(a_1) \leq \ell(a_2) \leq \ldots \leq \ell(a_{|\mathcal{S}|}) = \ell_{\max}$. Now, let us begin with a rooted q-ary tree whose all leaves are at depth $\ell_{\max}$. For a_1, we search for the first node at depth $\ell(a_1)$, remove all its children nodes to make it a leaf, and label it as $f(a_1)$; then for a_2, we search for the first node at depth $\ell(a_2)$, remove all its children nodes to make it a leaf, and label it as $f(a_2)$; we repeat this procedure for $a_3, a_4, \ldots$ until $a_{|\mathcal{S}|}$. In the above procedure, when labeling $f(a_i)$, we remove $q^{\ell_{\max}-\ell(a_i)}$ leaves in the original fully-grown tree. Since $\{\ell(s) : s \in \mathcal{S}\}$ satisfies (5.21), the procedure is guaranteed to be accomplished, and it produces a prefix-free code for S. □

Remark 5.3 From the proof, we see that the Kraft inequality attaining equality is equivalent to the corresponding rooted q-ary tree having no unused leaf.

We may wonder whether it is possible to have less stringent restriction on the lengths of index strings if we consider uniquely decodable codes that are not necessarily prefix-free. The following theorem excludes such a possibility.

Theorem 5.2 For a DMS S with $|\mathcal{S}| < \infty$, any of its uniquely decodable codes must satisfy the Kraft inequality (5.21) too.

Proof: Let us take the k-th power of the left hand side of (5.21), and expand it as

$$\begin{aligned}\left(\sum_{s\in\mathcal{S}} q^{-\ell(s)}\right)^k &= \sum_{s_1,s_2,\ldots,s_k\in\mathcal{S}^k} q^{-\ell(s_1)-\ell(s_2)-\ldots-\ell(s_k)} \\ &= \sum_{\underline{s}\in\mathcal{S}^k} q^{-\ell(\underline{s})}, \qquad (5.23)\end{aligned}$$

where $\underline{s} = [s_1, s_2, \ldots, s_k]$ and $\ell(\underline{s})$ is exactly the length of $f(\underline{s})$. We can alternatively rewrite the summation (5.23) by enumerating $\ell(\underline{s})$, which takes value from k to $k\ell_{\max}$, as

$$\sum_{\underline{s}\in\mathcal{S}^k} q^{-\ell(\underline{s})} = \sum_{m=k}^{k\ell_{\max}} A(m)q^{-m}, \qquad (5.24)$$

where $A(m)$ denotes the number of length-k source messages whose index strings have length m. Since the code is uniquely decodable, we have $A(m) \leq q^m$. Therefore,

$$\sum_{m=k}^{k\ell_{\max}} A(m)q^{-m} \leq \sum_{m=k}^{k\ell_{\max}} q^m q^{-m} = k(\ell_{\max}-1). \qquad (5.25)$$

So in summary, we have

$$\sum_{s\in\mathcal{S}} q^{-\ell(s)} \leq [k(\ell_{\max}-1)]^{1/k} \longrightarrow 1, \qquad (5.26)$$

as $k \longrightarrow \infty$. This completes the proof. □

Remark 5.4 A key consequence of Theorem 5.2 is that, for any uniquely decodable code, there always exists a prefix-free code that have exactly the same collection of index string lengths. Therefore, in order to study the possible index string lengths of a source, it loses no generality to focus on prefix-free codes.

5.4 Shannon-Fano Code and Huffman Code

The Kraft inequality provides a fundamental constraint on the index string lengths of any uniquely decodable code, but it does not reflect the impact of the probability distribution of a source. Now, consider a DMS S, and any of its q-ary uniquely decodable code with lengths $\{\ell(s) : s \in \mathcal{S}\}$. Let us calculate the relative entropy between P_S and another probability distribution Q defined as

$$Q(s) = \frac{q^{-\ell(s)}}{\sum_{s'\in\mathcal{S}} q^{-\ell(s')}}, \quad s \in \mathcal{S}, \qquad (5.27)$$

to obtain

$$\begin{aligned} D(P_S \| Q) &= \sum_{s\in\mathcal{S}} P_S(s) \log_q \frac{P_S(s)}{Q(s)} \\ &= -H(S) - \sum_{s\in\mathcal{S}} P_S(s) \log_q \frac{q^{-\ell(s)}}{\sum_{s'\in\mathcal{S}} q^{-\ell(s')}} \\ &= -H(S) - \sum_{s\in\mathcal{S}} P_S(s) \left[-\ell(s) - \log_q \left(\sum_{s'\in\mathcal{S}} q^{-\ell(s')} \right) \right] \\ &= -H(S) + \sum_{s\in\mathcal{S}} P_S(s)\ell(s) + \log_q \left(\sum_{s'\in\mathcal{S}} q^{-\ell(s')} \right) \\ &\leq -H(S) + \bar{\ell}, \end{aligned} \tag{5.28}$$

where the last inequality is due to the Kraft inequality. From the nonnegativity of relative entropy (see Theorem 3.4), we immediately have that any uniquely decodable code should satisfy $\bar{\ell} \geq H(S)$ (in base q). Furthermore, equality holds if and only if $\ell(s) = -\log_q P_S(s), \forall s \in \mathcal{S}$. While the index string length $\ell(s)$ should be an integer for every $s \in \mathcal{S}$, it is generally not true that $-\log_q P_S(s)$ is an integer for every $s \in \mathcal{S}$. If for a DMS this happens to be the case, such a DMS is called q-adic, and there exist prefix-free codes that exactly achieve the lower bound $H(S)$ (in base q) on the expected index string length.

Example 5.2 Suppose that a DMS S has $P_S(a_1) = 1/2, P_S(a_2) = 1/4$, and $P_S(a_3) = P_S(a_4) = 1/8$. When $q = 2$, we have $-\log_2 P_S(a_1) = 1$, $-\log_2 P_S(a_2) = 2$, and $-\log_2 P_S(a_3) = -\log_2 P_S(a_4) = 3$. So S is 2-adic. But it is not q-adic for any $q > 2$.

When a DMS is not q-adic, it is impossible to assign a non-integer value $-\log_q P_S(s)$ to $\ell(s)$. A simple and reasonable idea is to round up $-\log_q P_S(s)$ to get an integer-valued index string length; that is, for a DMS S, let $\ell(s) = \lceil -\log_q P_S(s) \rceil, \forall s \in \mathcal{S}$. Here $\lceil t \rceil$ is the smallest integer no less than t. Clearly, this assignment satisfies the Kraft inequality, and the corresponding prefix-free code is called the Shannon-Fano code.

From $-\log_q P_S(s) \leq \ell(s) = \lceil -\log_q P_S(s) \rceil < -\log_q P_S(s) + 1$, we have that the expected index length of the Shannon-Fano code satisfies

$$H(S) \leq \bar{\ell} < H(S) + 1, \tag{5.29}$$

where $H(S)$ is in base q. Similar to the last part of Section 5.2, when applying the Shannon-Fano code to a length-n source message $\underline{S}$, the rate $R = \bar{\ell}/n$ satisfies

$$H(S) \leq R < H(S) + \frac{1}{n}, \tag{5.30}$$

and therefore $R \longrightarrow H(S)$ as $n \longrightarrow \infty$. Again, we see that the fundamental performance limit of exact lossless compression is $H(S)$, even under the restriction of unique decodability.

Remark 5.5 Interestingly, here the rate R approaches $H(S)$ from above, while in Section 5.2 the rate R approaches $H(S)$ from below. This slight difference reflects the extra overhead due to the constraint of unique decodability.

Despite of its simplicity, the Shannon-Fano code is suboptimal in general. Below is an example.

Example 5.3 Consider a Bernoulli DMS S with parameter $1/16$. The binary Shannon-Fano code has index string lengths 1 and 4. Clearly, it is wasteful to use an index string of length 4, because for $|\mathcal{S}| = 2$ the we can simply let the code consist of $\{0, 1\}$, — both of length 1.

Next we introduce the Huffman code, which is the optimal prefix-free code in the sense that it minimizes the expected index string length.

Let us consider the binary case ($q = 2$) first. Assume that $K = |\mathcal{S}| < \infty$. Without loss of generality, we rearrange the source messages so that $P_S(a_1) \geq P_S(a_2) \geq \ldots \geq P_S(a_K)$.

For an optimal prefix-free code, consider its binary tree visualization. The following two facts hold.

1. All leaves in the binary tree correspond to index strings.
2. The index strings can always be arranged so that those of the two least likely source messages are siblings at the maximum depth in the binary tree, i.e., they differ only in their last position.

Both facts can be shown by contradiction, and are left as exercise.

So the expected index string length of an optimal prefix-free code can be written as follows:

$$\begin{aligned}
\bar{\ell} &= \sum_{s \in \mathcal{S}} P_S(s)\ell(s) \\
&= \sum_{s \in \mathcal{S} \setminus \{a_{K-1}, a_K\}} P_S(s)\ell(s) + P_S(a_{K-1})\ell(a_{K-1}) + P_S(a_K)\ell(a_K) \\
&= \sum_{s \in \mathcal{S} \setminus \{a_{K-1}, a_K\}} P_S(s)\ell(s) + [P_S(a_{K-1}) + P_S(a_K)]\ell(a_{K-1}). \quad (5.31)
\end{aligned}$$

This suggests that we can "merge" the leaves for $f(a_{K-1})$ and $f(a_K)$, by converting their parent node into a leaf replacing them, corresponding to a new message occurring with probability $P_S(a_{K-1}) + P_S(a_K)$. Accordingly, we obtain a new DMS S', with alphabet $\mathcal{S}' = \{a_1, a_2, \ldots, a_{K-2}, a'_{K-1}\}$, $P_{S'}(s') = P_S(s')$ for $s' \in \{a_1, a_2, \ldots, a_{K-2}\}$

and $P_{S'}(a'_{K-1}) = P_S(a_{K-1}) + P_S(a_K)$. Furthermore, the binary tree for S after the merging operation corresponds to a prefix-free code for S'.

Then we can rewrite (5.31) as

$$\bar{\ell} = \sum_{s' \in \mathcal{S}'} P_{S'}(s')\ell(s') + [P_S(a_{K-1}) + P_S(a_K)], \tag{5.32}$$

in which the summation on the right hand side is the expected index string length for the code of S'.

In order for $\bar{\ell}$ to be minimized, the prefix-free code for S' should be optimal and its corresponding binary tree should satisfy Facts 1 and 2, as well. So we can repeat the preceding argument, to further merge the two least likely source messages in $\mathcal{S}'$. Continuing this procedure recursively, until no more merging is possible, i.e., obtaining a source with only one source message, we obtain a binary tree that corresponds to the optimal prefix-free code of S.

Then consider the general q-ary case. For the optimal prefix-free code, let us inspect Facts 1 and 2 regarding its q-ary tree visualization.

These two facts now become the following: We can arrange the index strings so that the corresponding q-ary tree has at most $q - 2$ unused leaves, all of which are located at the maximum depth, in the same branch grown by a single parent node.

The "sort and merge" argument in the binary case still works, but some modification becomes necessary in the first step, because the number of leaves to be merged at the maximum depth may be fewer than q. Indeed, we need to determine the number of unused leaves $0 \leq r \leq q - 2$ at the maximum depth. For this, note that the total number of leaves in a finite q-ary tree can always be written as $q + n(q - 1)$ where n is an integer corresponding to the number of non-leaf nodes (excluding the root). Hence we have

$$\begin{aligned} K + r &= q + n(q-1) & (5.33)\\ \text{i.e.,} \quad q - K &= -n(q-1) + r, & (5.34) \end{aligned}$$

which is a remainder problem. We can thus obtain r as the remainder when dividing $q - K$ by $q - 1$. For convenience, we add $(K - q)(q - 1)$ to both sides, to get

$$(K-q)(q-2) = (K-q-n)(q-1) + r. \tag{5.35}$$

This determines r the number of unused leaves at the maximum depth. In subsequent steps, there is no unused leaf and we always merge q leaves until reaching the root.

In summary, the Huffman q-ary tree is generated as follows.

1. Create K nodes as $u_1, u_2, \ldots, u_K$, and assign to u_k the probability $P_S(a_k)$, $k = 1, \ldots, K$. Initialize these nodes as "active". Calculate r as the remainder when dividing $(K - q)(q - 2)$ by $q - 1$. For $q = 2$, r is always zero.
2. Link together the $q - r$ active nodes with the smallest probabilities to create a new node. Mark the $q - r$ linked nodes as "inactive" and the new node as "active". Assign to the new node the sum of the probabilities of the $q - r$ linked nodes.
3. If there is only one active node left, mark it as the root and stop, otherwise, set $r = 0$ and repeat Step 2.

Having generated the Huffman q-ary tree, the corresponding prefix-free code immediately follows, which minimizes the expected index string length.

Note that the Huffman code is applicable to the case of $|\mathcal{S}| < \infty$ only. If $|\mathcal{S}| = \infty$, it is generally unclear what the optimal code looks like. Nevertheless, for a DMS obeying the geometric distribution, the optimal code is known; see [24].

Notes

In our lecture notes this lecture on lossless compression is positioned largely as a special case of the general theory of source representation after Lecture 4. Historically, however, the study of exactly lossless compression preceded that of the rate-distortion theory. The Kraft inequality first appeared in Leon Kraft's master's thesis at MIT around 1949 [25], for prefix-free codes. Its more general form for uniquely decodable codes in Theorem 5.2 was discovered by Brockway McMillan [26], and its proof in this lecture was found by Jack Karush [27]. The Shannon-Fano code was first implicitly described in Shannon's original article [1] and was independently discovered by Robert Fano. The construction of optimal code for lossless compression was initially thought to be an intractable problem, but was solved by David Huffman in 1951 [28], in a term paper assigned by Robert Fano in his MIT information theory class. Our explanation of the Huffman code is based on James L. Massey's lecture notes [29]. Lossless compression is also closely related to random number generation; see, e.g., [18, Chap. 5.11].

Due to the great importance of reliability in computer file systems, numerous lossless compression codes have been invented. Unlike Huffman codes which encode a fixed-length source symbol sequence into a variable-length string, Tunstall codes [30] encode a variable-length source symbol sequence into a fixed-length string. Arithmetic codes, which was built upon the Shannon-Fano-Elias codes, have been widely used; see, e.g., [18, Chap. 13.3].

Run-length codes [31] are very efficient for compressing source symbol sequences with long subsequences of all-zeros and all-ones. Burrows-Wheeler transform [32] is a clever way of rearranging a source symbol sequence to yield long runs and can be used as a preprocessing step for run-length coding. A widely used class of codes in practice is called universal codes in the sense that they require no knowledge on the source probability distribution, and asymptotically achieve the optimal compression efficiency (the entropy for DMS, and the entropy rate for stationary or even more general sources) when the length of the source symbol sequence grows without bound. Lempel-Ziv codes have been perhaps the most well known universal coding scheme to date, invented by Abraham Lempel and Jacob Ziv in the 1970s.

Exercises

1. For a DMS S under distortion measure $d(s, \hat{s})$, define a new distortion measure as $\tilde{d}(s, \hat{s}) = 1$ if $d(s, \hat{s}) > a$ and 0 otherwise, given some parameter $a > 0$. Describe the rate-distortion function of S under $\tilde{d}$ at $D = 0$.
2. In Section II, we have studied exactly lossless compression when the index string is binary. If the index string is q-ary, $q \geq 2$, derive lower and upper bounds on the expected index string length, by generalizing the analysis in Section II.
3. For the exactly lossless compression code studied in Section II, numerically evaluate $\bar{\ell}$ when S is (a) uniform over $\{1, 2, \ldots, M\}$, and (b) geometric with parameter ϵ. Compare the exact values of $\bar{\ell}$ under these cases with the upper and lower bounds obtained in Section II.
4. Prove that a code is uniquely decodable if and only if for any integer $n \geq 1$, and any $\underline{s} \neq \underline{s}' \in \mathcal{S}^n$, $f(\underline{s}) \neq f(\underline{s}')$.
5. Instead of (5.21) in Theorem 5.1, we may rewrite the Kraft inequality as

$$\sum_{\ell=1}^{\infty} A_\ell q^{-\ell} \leq 1, \tag{5.36}$$

where A_ℓ denotes the number of index strings of length ℓ. Let us use this form of the Kraft inequality to prove the converse part of Theroem 5.1; that is, for a given set of $\{A_\ell : \ell = 1, 2, \ldots\}$ satisfying the Kraft inequality, we can construct a corresponding prefix-free code. Starting with the root node, complete the following induction:

   a) Prove that from the root node, there are at least A_1 leaves at depth 1 to accommodate the A_1 length-1 index strings.

   b) Suppose that we have already accommodated all index strings from length 1 to length $\ell - 1$. Prove that there are at least A_ℓ unused leaves at depth ℓ to accommodate the A_ℓ length-ℓ index strings.

6. A code is called suffix-free if for any $s \neq s' \in \mathcal{S}$, $f(s)$ is not a suffix of $f(s')$, and is called fix-free if it is both prefix-free and suffix-free. For a DMS S with $|\mathcal{S}| < \infty$, when $\sum_{s\in\mathcal{S}} q^{-\ell(s)} \leq 1/2$, find a method to construct a q-ary fix-free code with lengths $\{\ell(s) : s \in \mathcal{S}\}$.
7. Prove that for a DMS S with $|\mathcal{S}| = \infty$, a prefix-free code still satisfies the Kraft inequality, and conversely, for any index string lengths satisfying the Kraft inequality there exists a corresponding prefix-free code. (Hint: Instead of using a rooted tree visualization, visualize a prefix-free code as a partitioning of the unit interval $[0, 1]$, such that each index string corresponds to a sub-interval, and all the sub-intervals are disjoint; see [18, Chap. 5.5])
8. Derive the binary Huffman code for a DMS S uniformly distributed over $\{1, 2, \ldots, 10000\}$, and compare the resulting expected index string length with the entropy bound $\log_2 10000$ bits.
9. For a DMS S, we design a prefix-free code that minimizes the weighted expected index string length $\bar{\ell} = \sum_{s\in\mathcal{S}} P_S(s)c(s)\ell(s)$, where $c(s) > 0$ is the cost per index position for source message s. Note that when $c(s) = 1$, $\forall s \in \mathcal{S}$, we return to the problem studied in Section IV and it is solved by the Huffman code there.

   a) Derive a lower bound on $\bar{\ell}$, and discuss when this lower bound can be achieved.
   b) Generalize the Huffman code to yield the prefix-free code that minimizes $\bar{\ell}$.

10. For a DMS S with K positive probabilities and one zero probability, i.e., $P_S(a_1) \geq P_S(a_2) \geq \ldots \geq P_S(a_K) > P_S(a_{K+1}) = 0$, we may either design a Huffman code omitting the zero probability, or including it. Find the relationship between the expected index string lengths of these two different Huffman codes.
11. Consider independent DMSs S_1 and S_2 with (not necessarily identical) finite alphabets. Denote their binary Huffman codes as f_{S_1} and f_{S_2}, respectively. Now view (S_1, S_2) as a single DMS, and use the concatenation $[f_{S_1}, f_{S_2}]$ as the code for (S_1, S_2); for example, if $f_{S_1}(s_1) = 001$ and $f_{S_2}(s_2) = 101$ for some (s_1, s_2), then $f_{S_1,S_2}(s_1, s_2) = 001101$.

   a) Show that f_{S_1,S_2} is a prefix-free code.
   b) Does the Kraft inequality for f_{S_1,S_2} always hold equal?

12. The Shannon-Fano code adopts a conservative philosophy

by rounding up all non-integer values of $-\log_q P_S(s)$. It may be possible to judiciously round down some non-integer values of $-\log_q P_S(s)$, so as to obtain a prefix-free code with a smaller expected index string length.

a) Propose an algorithm for designing a prefix-free code that may outperforms the Shannon-Fano code, by selectively rounding down some non-integer values of $-\log_q P_S(s)$.

b) Find an example where your designed code is strictly worse than the Huffman code.

13. In the problem formulation of lossy source representation in Lecture 4, the encoded index $W \in \{1, 2, \ldots, M_n\}$ may also be viewed as a binary string of a fixed length $\lceil \log_2 M_n \rceil$. Now, if we allow W to be of variable length, drawn from the set of all finite-length binary strings $\mathcal{W}^* = \{\emptyset, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$. Define the rate of a code by $R = \mathbf{E}[\ell(\underline{S})]/n$, where $\ell(\underline{S})$ is the length of W encoding $\underline{S}$ and n is the length of $\underline{S}$. Modify the proof of the converse part in Lecture 4, to show that variable-length coding still cannot outperform the rate-distortion function.