Chapter 2

Variable Length Source Coding

2.1 Source Coding Problems

Consider that you are asked to store the text of book in a computer. What is the best you can do in order to save the storage space? Similar problems exist for audio voices and video clips, where though the original information is continuous, audio and video can be sampled and quantized and hence represented by digital data.

This kind of problems are called *source coding* and can be modelled as follows: Fix a finite alphabet $\mathcal{A}$, whose elements, for example, can be $a, b, c, \ldots$, or $0, 1, \ldots$. You are given a sequence $\mathbf{x}$ of n symbols

$$x_1, x_2, \ldots, x_n, \quad x_i \in \mathcal{A}$$

to be stored in a computer as a sequence of bits $\mathbf{c}$. What is the minimum number of bits in $\mathbf{c}$ so that you can later recover the sequence $\mathbf{x}$ from $\mathbf{c}$? The process to generate $\mathbf{c}$ from $\mathbf{x}$ is called *encoding* and the reverse process to obtain $\mathbf{x}$ from $\mathbf{c}$ is called *decoding*.

2.1.1 Variable-length Codes

If we exam all the entries of $\mathbf{x}$, we can see that different letters in $\mathcal{A}$ may not appear the same times, and hence we can opt to a shorter representation for a letter appears more frequently. The idea may date back to Morse code or earlier.

Suppose that $\mathcal{A} = \{a, b, c, d\}$, and let $\mathbf{x}$ be

$$a\ a\ a\ b\ a\ c\ a\ d\ b\ c\ b\ d\ c\ d\ d\ a\ b\ c\ b\ d\ c\ d\ d\ a\ a\ a\ a\ b\ a\ c\ a\ d \tag{2.1}$$

where a, b, c, d appears $12, 6, 6, 8$ times, respectively. Can we use $0, 1, 01, 10$ to represent a, b, c, d, respectively? With this representation, the encoding of *aaab* would generate sequence

$$0001$$

We see that the encoding of *aac* also generates the same sequence. So this representation of a, b, c, d cannot guarantee the decoding is unique.

We will study this approach more carefully later in this chapter to see what is a valid representation of letters in $\mathcal{A}$ using variable-length bit sequences, and how to find such a representation.

2.1.2 Block Codes

Another approach to represent the sequence $\mathbf{x}$ in bits is to assign the same number of bits to represent each symbol in $\mathcal{A}$. Suppose that $\mathcal{A} = \{a, b, c, d\}$. Each letter of $\mathcal{A}$ can be represented by two bits uniquely (which is already shortest), and hence $\mathbf{x}$ can be encoded to $2n$ bits.

Can we do better than $2n$ bits? We can try to represents to consecutive bits in $\mathbf{x}$ jointly. There are totally 16 different combinations of two letters in $\mathcal{A}$, each of which can be represented by 4 bits uniquely (which cannot be shorter). But this does not save any bits.

Suppose that our sequence $\mathbf{x}$ come from a language using the 4 letters in $\mathcal{A}$ to form words of two letters. But not all the combinations of two letters are valid words: there are only eight valid words in the language. Due to this fact, we only need to give a unique representation for these eight words, which can be done using 3 bits. Hence, $\mathbf{x}$ can be encoded to $1.5n$ bits.

Exploring the higher order of structures, e.g., sentences and paragraphs of this language, we can further reduce the length of the encoding.

2.2 Variable-Length Source Coding

Variable-Length Source Coding

- Let $\mathcal{X}^*$ be the set of finite length strings of symbols from an alphabet $\mathcal{X}$.
- Let $\mathcal{B} = \{0, 1\}$.
- A binary (variable-length) source code for an alphabet $\mathcal{A}$ is a mapping
$$f : \mathcal{A} \to \mathcal{B}^*.$$
- The encoding of a *message sequence* $\mathbf{x}$ of n symbols from $\mathcal{A}$ is the *coded sequence*
$$f(\mathbf{x}) = f(x_1)f(x_2)\cdots f(x_n)$$
- The decoding of $\mathbf{c} \in \mathcal{B}^*$ is a function $\varphi : \mathcal{B}^* \to \mathcal{A}^*$.

Performance Measure

- Let $l(\mathbf{c})$ is the length of a sequence $\mathbf{c} \in \mathcal{B}^*$.
- The average length of the coded sequence of $\mathbf{x}$ is
$$L_f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i)).$$
- Denote by $N(a|\mathbf{x})$ the number of occurrences of a in $\mathbf{x}$. We can write
$$L_f(\mathbf{x}) = \sum_{a\in\mathcal{A}} \frac{N(a|\mathbf{x})}{n} l(f(a)).$$
- Let $l_a = l(f(a))$, $a \in \mathcal{A}$.

Uniquely decodable codes

Definition 2.1 A code f is *uniquely decodable* if for any $\mathbf{x}, \mathbf{x}' \in \mathcal{A}^*$, we have $f(\mathbf{x}) \neq f(\mathbf{x}')$.

■ **Example 2.1** Let $\mathcal{X} = \{A, B, C, D\}$. Consider a code defined by

$$A \mapsto 0, \quad B \mapsto 1$$
$$C \mapsto 01, D \mapsto 10.$$

Then all the three source sequences AAD, ACA, and $AABA$ produce the code sequence 0010. Therefore, the code is not uniquely decodable. ■

Kraft Inequality

Theorem 2.1 — Kraft Inequality. For a source alphabet $\mathcal{A}$, a binary uniquely decodable code f satisfies
$$\sum_{a\in\mathcal{A}} 2^{-l(f(a))} \leq 1. \tag{2.2}$$

Proof. For any integer $N > 0$, write
$$\left(\sum_{a\in\mathcal{A}} 2^{-l_a}\right)^N = \sum_{a_1\in\mathcal{A}}\sum_{a_2\in\mathcal{A}}\cdots\sum_{a_N\in\mathcal{A}} 2^{-(l_{a_1}+l_{a_2}+\ldots+l_{a_N})} = \sum_{i=1}^{N_{\max}} A_i 2^{-i}$$
where A_i is the number of N-length source sequences with coded sequence length i. As the code is uniquely decodable, $A_i \leq 2^i$. The proof is completed by $N_{\max} \leq N \max_{a\in\mathcal{A}} l_a$. ■

Types

- Denote by $N(a|\mathbf{x})$ the number of occurrences of a in $\mathbf{x}$.
- The *type* of a sequence $\mathbf{x} \in \mathcal{X}^n$ is the distribution $P_{\mathbf{x}}$ on $\mathcal{X}$ defined by

$$P_{\mathbf{x}}(a) = \frac{1}{n} N(a|\mathbf{x}) \quad \text{for every} \quad a \in \mathcal{X}.$$

- A type is a PMF.
- The average coded sequence length of $\mathbf{x}$ depends only on the type of $\mathbf{x}$.
- For any PMF p on $\mathcal{A}$, define the expected codeword length of a code f for $\mathcal{A}$ as

$$L_f(p) = \sum_{a \in \mathcal{A}} p(a) l_a.$$

Entropy Bound

Theorem 2.2 Consider a binary uniquely decodable code f and any distribution p on $\mathcal{A}$,

$$L_f(p) \geq H(p)$$

This lower bound is tight if and only if $l(f(a)) = -\log p(a)$.

Proof.

$$\begin{aligned} L_f(p) - H(p) &= \sum_{a \in \mathcal{A}} p(a) \log 2^{l_a} + \sum_{a \in \mathcal{A}} p(a) \log P_{\mathbf{x}}(a) \\ &= \sum_{a \in \mathcal{A}} p(a) \log \frac{p(a)}{2^{-l_a}} \geq \log \frac{1}{\sum_a 2^{-l_a}} \geq 0, \end{aligned}$$

where the first inequality follows from the log-sum inequality and the second inequality follows from the Kraft inequality. ■

2.3 Type

For a general type P of sequences from $\mathcal{A}^n$, define the set of sequences of type P as T_P^n. A uniquely decodable code f on $\mathcal{A}$ generate the same average coded sequence length $L_f(P)$ for all sequences of the same type P, and hence, we must have $|T_P^n| \leq 2^{nL_f(P)}$.

■ **Example 2.2** Let $\mathcal{A} = \{1, 2, 3\}$ and $\mathbf{x} = 11321$. Show $P_{\mathbf{x}}$, $T_{P_{\mathbf{x}}}$ and $|T_{P_{\mathbf{x}}}|$. ■

Lemma 2.3 — Type counting. The number of different types of sequences in $\mathcal{A}^n$ is less than $(n+1)^{|\mathcal{A}|}$.

Proof. A type can be given by a $|\mathcal{X}|$-tuple, where each component can take $n+1$ different values. ■

R The number of sequences in $\mathcal{A}^n$ is exponential in n, but the number of different types of sequences in $\mathcal{A}^n$ is at most polynomial in n.

Number of Sequences of a Type

- Consider a type $P = (P_1, \ldots, P_M)$ of length-n sequences.
- $|T_P^n| = \binom{n}{nP_1, nP_2, \ldots, nP_M}$.
- $|T_P^n| \approx 2^{n(H(P)+o(n))}$.

Lemma 2.4 For any type P of sequences in $\mathcal{A}^n$,

$$(n+1)^{-|\mathcal{A}|} 2^{nH(P)} \leq |T_P^n| \leq 2^{nH(P)}. \tag{2.3}$$

Proof. The lemma can be proved using Stirling's approximation to bound the factorial function. ■

2.4 Prefix codes

- We will show the existence of a uniquely decodable code f for distribution p with $L_f(p) < H(p) + 1$.
- Hence, for a message sequence $\mathbf{x}$, the minimum average codeword length $L^*(\mathbf{x})$ satisfies

$$H(P_{\mathbf{x}}) \leq L^*(\mathbf{x}) < H(P_{\mathbf{x}}) + 1.$$

Prefix Codes

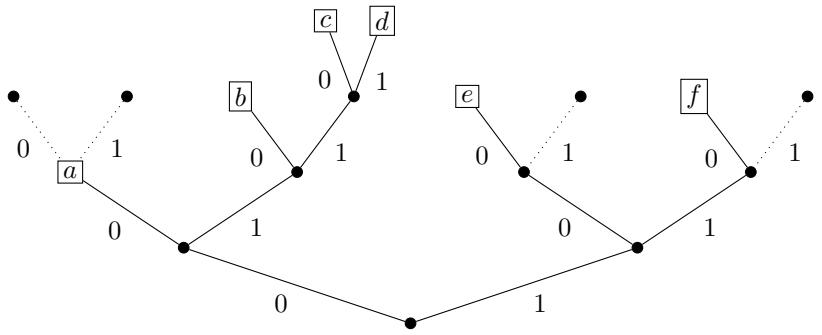
- A code f for $\mathcal{A}$ is said to be *prefix-free* if no codeword $f(a)$ is a prefix of any other codeword $f(a')$, $a \neq a'$.
- For brevity, a prefix-free code will be called a *prefix code*.
- A prefix code is uniquely decodable.
- Example:

$$A \mapsto 0$$
$$B \mapsto 10$$
$$C \mapsto 110$$
$$D \mapsto 1111$$

Code Tree

- A binary prefix code can be represented by a binary tree with the leaves being the codewords.
- The prefix code corresponding the code tree in the figure is

$$a \mapsto 00, \quad b \mapsto 010, \; c \mapsto 0110,$$
$$d \mapsto 0111, \; e \mapsto 100, \; f \mapsto 110.$$



Because a prefix code is uniquely decodable and hence satisfies Kraft's inequality.

Existence

Theorem 2.5 For any integers l_a, $a \in \mathcal{A}$ such that the Kraft inequality $\sum_{a\in\mathcal{A}} 2^{-l_a} \leq 1$ is satisfied, there exists a binary prefix code f for $\mathcal{A}$ such that $l(f(a)) = l_a$.

Proof. Denote that $\mathcal{A} = \{1, \ldots, m\}$. Suppose that $1 \leq l_1 \leq l_2 \leq \ldots \leq l_m$ satisfy the Kraft inequality. Construct a code by associating each $i \in \mathcal{A}$ to nodes in the binary coding tree so that $f(i)$ takes the binary sequence specifying node.

In the ith step, we associate a node in the tree of depth l_i to i, which should not be the children of the nodes associated to $1, \ldots, i-1$. As the nodes associated to $1, \ldots, i-1$ do not share the same children, we have there are at most $\sum_{k=1}^{i-1} 2^{l_i - l_k} < 2^{l_i}$ nodes that cannot be associated to i, where the inequality follows from the Kraft inequality and $2^{-l_i} > 0$. ■

See also [7, Theorem 4.11] for the existence of such a prefix code.

Upper bound

Theorem 2.6 For any distribution p on $\mathcal{A}$, there exists a prefix code f for $\mathcal{A}$ such that

$$L_f(p) < H(p) + 1.$$

Proof.
- Let $l_a = \lceil -\log_2 p(a) \rceil$.
- First, $\{l_a\}$ satisfies the Kraft inequality.
- Then, by the existence theory, there exists a binary prefix code with code lengths $\{l_a\}$.
- Last, $L_f(p) = \sum_a p(a) l_a < H(p) + 1$.

■

2.5 Huffman codes

An optimal (shortest expected length) prefix code for a given distribution (or type) can be constructed by a simple algorithm discovered by Huffman. We will prove that any other code for the same alphabet cannot have a lower expected length than the code constructed by the Huffman procedure.

Huffman Codes
- A Huffman code (tree) is constructed by the Huffman procedure which keeps merging the two smallest probability masses until one probability mass (i.e., 1) is left.
- Consider a distribution on the set $\mathcal{A} = \{1, 2, 3, 4, 5\}$ with probability masses $0.25, 0.25, 0.2, 0.15, 0.15$, respectively.

Example

$l(f(x))$	$f(x)$	x	$p(x)$			
2	01	1	0.25	0.3	0.45	0.55 — 1
2	10	2	0.25	0.25	0.3	0.45
2	11	3	0.2	0.25	0.25	
3	000	4	0.15	0.2		
3	001	5	0.15			

A source code f for $\mathcal{A}$ is said to be optimal with respect to a distribution p on $\mathcal{A}$ if $L_f(p) \leq L_{f'}(p)$ for any other code f for $\mathcal{A}$.

Lemma 2.7 For an optimal code f, if $p(a) \geq p(b)$ then $l(f(a)) \leq l(f(b))$.

Proof. If the claim does not hold, a better code exists. ■

Lemma 2.8 There exists an optimal code f such that for $a, b \in \mathcal{A}$ with $p(a), p(b)$ being the two smallest probability masses, $f(a)$ and $f(b)$ are siblings in the coding tree, i.e., they have the same length and they differ only in the last symbol.

Proof. First show that for an optimal code the sibling of the codeword assigned to the smallest probability must also be a codeword, since otherwise, we can replace the codeword by its parent in the coding tree. Second, using Lemma 2.7, show that the sibling can be assigned to the second smallest probability without lossing the optimality. ■

Theorem 2.9 The Huffman procedure generates an optimal prefix code for an alphabet $\mathcal{A}$ with respect to any distribution p.

Proof. Let $\mathcal{A}_m = \{1, \ldots, m\}$ and p_m be a PMF on $\mathcal{A}_m$ such that $p_m(1) \geq p_m(2) \geq \cdots \geq p_m(m)$, for $i = 1, \ldots, m-2$, $p_m(i) = p_{m-1}(i)$ and $p_{m-1}(m-1) = p_m(m-1) + p_m(m)$. Let f_m be an optimal uniquely decodable source code for p_m satisfying the property in Lemma 2.8. A code f_{m-1} for p_{m-1} can be constructed as follows: for $i = 1, \ldots, m-2$, $f_{m-1}(i) = f_m(i)$, and $f_{m-1}(i-1)$ is $f_m(m)$ (or $f_m(m-1)$) without the last symbol. It can be calculated that

$$L_{f_m}(p_m) - L_{f_{m-1}}(p_{m-1}) = p_m + p_{m-1}.$$

Let h_m and h_{m-1} be the Huffman codes for $\mathcal{A}_m$ and $\mathcal{A}_{m-1}$, respectively, with $h_m(i) = h_{m-1}(i)$ for $i = 1, \ldots, m-2$. Similarly,

$$L_{h_m}(p_m) - L_{h_{m-1}}(p_{m-1}) = p_m + p_{m-1} = L_{f_m}(p_m) - L_{f_{m-1}}(p_{m-1}).$$

If Huffman coding is optimal for probability distributions of length $m-1$, i.e., $L_{h_{m-1}}(p_{m-1}) \leq L_{f_{m-1}}(p_{m-1}$, then $L_{h_{m-1}}(p_{m-1}) \leq L_{f_m}(p_m)$, i.e., Huffman coding is optimal for probability distributions of length m.

The proof is completed by induction. ■