Chapter 3 Source Coding

- 3.1 An Introduction to Source Coding
- 3.2 Optimal Source Codes
- 3.3 Shannon-Fano Code
- 3.4 Huffman Code

§ 3.1 An Introduction to Source Coding

- Entropy (e.g., in bits per symbol) implies the average number of bits that are required to represent a source symbol. This indicates a mapping between the source symbols and bits.
- **Source coding** can be seen as a mapping mechanism between source symbols and e.g., bits.
- For a string of symbols, how can we use less bits to represent them?

Intuition: Use short description to represent the most frequently occurred symbols; Use necessarily long description to represent the less frequently occurred symbols.

§ 3.1 An Introduction to Source Coding

Symbols:	1	2	4	4	3	1	4	4
	⇓							
bits:	00	01	11	11	10	00	11	11
	⇓							

Or can this be a shorter string of bits?

- **Expected Length :** Let x denote a source symbol and $C(x)$ denote a codeword of x. If the length of $C(x)$ is $l(x)$ (e.g., in bits) and x occurs with a probability of $p(x)$, the expected length $L(C)$ of source code C is:

$$L(C) = \sum_x p(x) \cdot l(x).$$

- It implies the average number of bits that are required to represent a source symbol in source coding scheme C.

§ 3.1 An Introduction to Source Coding

Let us look at the following example:

Example 3.1 Let X be a random variable with an alphabet of $\{1,\ 2,\ 3,\ 4\}$, it has a distribution of

$$P(x=1)=\frac{1}{2}, P(x=2)=\frac{1}{4}, P(x=3)=\frac{1}{8}, P(x=4)=\frac{1}{8}$$

Entropy of X is:

$$H(X)=\sum_{x\in\{1,2,3,4\}} P(x)\log_2 P(x)^{-1}$$

$$=1.75\ \text{bits/sym}.$$

§ 3.1 An Introduction to Source Coding

Source Coding 1 (C):

$$C(1) = 00,\ C(2) = 01,\ C(3) = 10,\ C(4) = 11$$

$$L(C) = \frac{1}{2}\cdot 2 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 2 + \frac{1}{8}\cdot 2 = 2 \text{ bits.}$$

On average, we use 2 bits to represent a symbol.

⇨ $L(C) > H(X)$.

Source Coding 2 (C^*):

$$C^*(1) = 0,\ C^*(2) = 10,\ C^*(3) = 110,\ C^*(4) = 111$$

$$L(C^*) = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 = 1.75 \text{ bits}$$

On average, we use 1.75 bits to represent a symbol.

⇨ $L(C^*) = H(X)$.

Observation: C^* should be a better source coding scheme than C.

§ 3.1 An Introduction to Source Coding

Memoryless Source: Given a source symbol sequence $s_1, s_2, \ldots, s_n$. It is memoryless if

$$P(s_j) = P(s_j \mid s_1, s_2, \ldots, s_{j-1}), \forall j = 1, 2, \ldots, n.$$

The source symbols are statistically independent.

Theorem 3.1 Shannon's Source Coding Theorem Given a memoryless source *X* whose symbols are chosen from the alphabet $\{x_1, x_2, \ldots, x_U\}$ with the alphabet symbol probabilities of $P(x_1) = p_1, P(x_2) = p_2, \ldots, P(x_U) = p_U$, and $\sum_{i=1}^{U} p_i = 1$. If the source is of length n, when $n \to \infty$, it can be encoded with $H(X)$ bits per symbol. The coded sequence will be of $nH(X)$ bits.

Note: $H(X) = \sum_{i=1}^{U} p_i \log_2 p_i^{-1}$ bits/sym.

§ 3.1 An Introduction to Source Coding

Important Features of Source Coding:

1. **Non-singularity:** Unambiguous representation of source symbols.
   That says if $x_i \neq x_j$, $c(x_i) \neq c(x_j)$.

X	$C(X)$
1	0
2	010
3	01
4	10

Problem: When we try to decode '010', it can be 2 or 14 or 31.

The decoding is NOT unique.

2. **Uniquely decodable:** A codeword can only be uniquely decoded into a source symbol.

X	$C(X)$
1	10
2	00
3	11
4	110

Problem: When we try to decode '001011000', we have

2 1 ↗ 3 2 . . .
2 1 ↘ 4 2

We will have to wait and see the end of the bit string. The decoding is NOT instantaneous.

§ 3.1 An Introduction to Source Coding

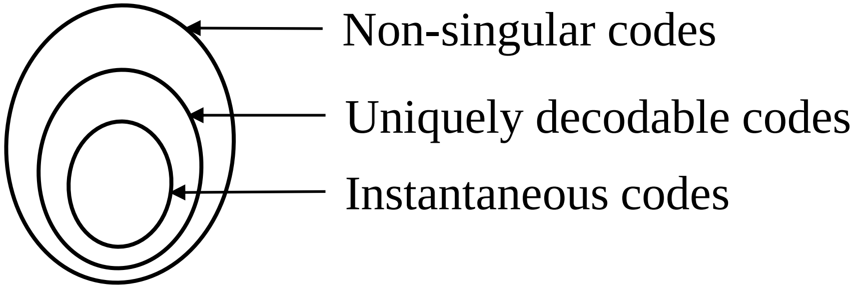
3. Instantaneous decoding: The decoding (demapping) happens once a codeword is read.

Instantaneous codes: For an instantaneous code, no codeword is a prefix of any other codeword.

X	$C(X)$
1	0
2	10
3	110
4	111

Observation: If you try to decode '111110101100111', you would notice that the puncturing positions are determined by the instances you have reached a source codeword. The decoding is instantaneous, and the decoding output is '4 3 2 3 1 4 ' .

Source Codes:



§ 3.2 Optimal Source Codes

How can we find an optimal source code?

An optimal source code :

(1) An instantaneous code (prefix code)

(2) The smallest expected length $L = \sum_i p_i l_i$

Theorem 3.2 Kraft Inequality For an instantaneous code over an alphabet of size D (e.g., $D = 2$ for binary codes), the codeword lengths $l_1, l_2, \cdots, l_U$ must satisfy

$$\sum_i D^{-l_i} \leq 1.$$

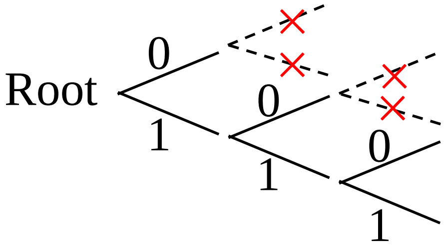
Remark: An instantaneous code $\rightleftarrows$ $\sum_i D^{-l_i} \leq 1$

Example 3.2 For the source code C^* of *Example 3.1*.

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1.$$

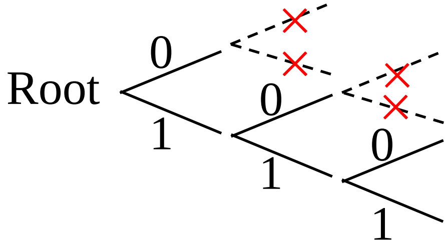
§ 3.2 Optimal Source Codes

Proof:



- Assume D = 2, and the above binary tree illustrates the assignment of binary source codeword. A complete solid path represents a source codeword.
- Based on property of instantaneous codes, if the first source codeword goes the '0' path, the next source codeword should not go the '0' path. Such a source codeword symbol assignment process repeats as the number of data symbols increases.

§ 3.2 Optimal Source Codes



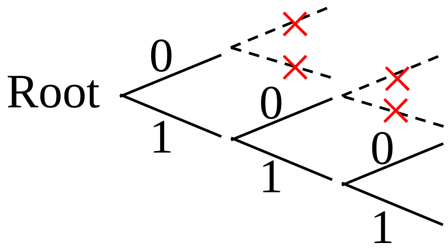
- At level $l_{\max}$ of the tree (source codeword length is $l_{\max}$), there are at most $D^{l_{\max}}$ codewords. Similarly, at level l_i of the tree, there are at most D^{l_i} codewords. A codeword at level l_i has $D^{l_{\max}-l_i}$ descendants at level $l_{\max}$.

- The descendent sets of all codewords should be disjoint. Consider all codewords, this property implies

$$\sum_i D^{l_{\max}-l_i} \leq D^{l_{\max}} \longrightarrow \sum_i D^{-l_i} \leq 1.$$

§ 3.2 Optimal Source Codes

- The tree represents an instantaneous source code.



- The expected length of this tree is

$$\mathbb{E}[l] = \sum_i l_i p_i$$

l_i: length of a source codeword for symbol x_i

p_i: probability of symbol x_i

- Expected length of the tree is the expected length of the source code.

§ 3.2 Optimal Source Codes

- Finding the smallest expected length L becomes

$$\text{minimize:} \quad L = \sum_i p_i l_i$$
$$\text{s.t.} \quad \sum_i D^{-l_i} \le 1.$$

- The constrained minimization problem can be interpreted through the Lagrange multipliers as:

$$\text{minimize:} \quad J = \sum_i p_i l_i + \lambda\left(\sum_i D^{-l_i}\right)$$

- Calculus: $\frac{\partial J}{\partial l_i} = p_i - \lambda D^{-l_i} \log_e D$. To enable $\frac{\partial J}{\partial l_i} = 0$, we need $D^{-l_i} = \frac{p_i}{\lambda \log_e D}$.

- To satisfy the Kraft Inequality, we have $\lambda = \frac{1}{\log_e D}$. Hence, $p_i = D^{-l_i}$.

- To minimized L, we need $l_i^* = \log_D p_i^{-1}$.

- With $l_i^* = \log_D p_i^{-1}$, we have

$$L = \sum_i p_i l_i^* = \sum_i p_i \log_D p_i^{-1} = H_D(X)$$

Entropy of the source symbols

§ 3.2 Optimal Source Codes

Theorem 3.3 (Lower Bound of the Expected Length) The expected length L of an instantaneous D-ary code for a random variable X is lower bounded by

$$L \geq H_D(X).$$

Proof:

$$\begin{aligned} L - H_D(X) &= \sum_i l_i p_i + \sum_i p_i \log_D p_i \\ &= -\sum_i p_i \log_D D^{-l_i} + \sum_i p_i \log_D p_i \\ &= \sum_i p_i \log_D \frac{p_i}{D^{-l_i}}. \end{aligned}$$

Let $p_i' = D^{-l_i}$,

$$\begin{aligned} L - H_D(X) &= \sum_i p_i \log_D \frac{p_i}{p_i'} \\ &= D(p_i || p_i') \geq 0. \end{aligned}$$

Remark: Since l_i can only be an integer,

$$L = H_D(X), \text{ if } l_i = -\log_D p_i.$$
$$L > H_D(X), \text{ if } l_i = \lceil -\log_D p_i \rceil.$$

§ 3.2 Optimal Source Codes

Corollary 3.4 (Upper Bound of the Expected Length) The expected length L of an instantaneous D-ary code for a random variable X is upper bounded by

$$L < H_D(X) + 1.$$

Proof: Since $-\log_D p_i \le l_i < -\log_D p_i + 1.$

By multiplying p_i to the above inequality and performing summation over i as

$$\sum_i -p_i \log p_i \le \sum_i p_i l_i < \sum_i -p_i \log p_i + \sum_i p_i$$

$$H_D(X) \le L < H_D(X) + 1.$$

§ 3.3 Shannon-Fano Code

- Given a source that contains symbols $x_1, x_2, \ldots, x_U$ with probabilities of $p_1, p_2, \ldots, p_U$, respectively.
- Determine the source codeword length for symbol x_i as

$$l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil \text{ bits.}$$

- Further determine $l_{\max} = \max\{l_i, \forall i\}$.
- **Shannon-Fano Code Construction**:

Step 1: Construct a binary tree of depth $l_{\max}$.

Step 2: Choose a node of depth l_i and delete its following paths and nodes. The path from root to the node represents the source codeword for source symbol x_i.

§ 3.3 Shannon-Fano Code

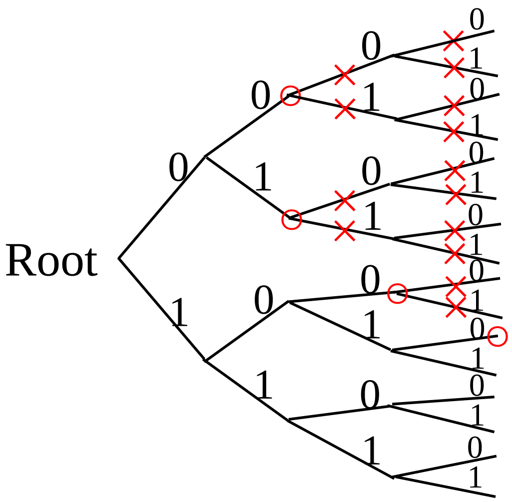
- ***Example 3.3*** Given a source with symbols x_1, x_2, x_3, x_4, they occur with a probability of $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.2$, $p_4 = 0.1$, respectively. Construct its Shannon-Fano code.

  We can determine

$$l_1 = \left\lceil \log_2 \frac{1}{p_1} \right\rceil = 2, l_2 = \left\lceil \log_2 \frac{1}{p_2} \right\rceil = 2, l_3 = \left\lceil \log_2 \frac{1}{p_3} \right\rceil = 3, l_4 = \left\lceil \log_2 \frac{1}{p_4} \right\rceil = 4,$$

  and $l_{\max} = 4$.

  Construct a binary tree of depth 4.



The source codewords are

x_1: 0 0

x_2: 0 1

x_3: 1 0 0

x_4: 1 0 1 0.

Note: $L = 2.4$ bits/sym., $H(X) = 1.85$ bits/sym.,

and $H(X) < L < H(X) + 1$.

§ 3.4 Huffman Code

- Given a source that contains symbols $x_1, x_2, \ldots, x_U$ with probabilities of $p_1, p_2, \ldots, p_U$, respectively.

- **Huffman Code Construction**:

  Step 1: Merge the 2 smallest symbol probabilities;

  Step 2: Assign the 2 corresponding symbols with 0 and 1, then go back to **Step 1**;

  Repeat the above process until two probabilities are merged into a probability of 1.

- Huffman code is the shortest prefix code, i.e., an optimal code.

§ 3.4 Huffman Code

Example 3.4 Given a source with symbols x_1, x_2, x_3, x_4, x_5. They occur with probabilities of $P_1 = 0.25,\ P_2 = 0.25, P_3 = 0.2,\ P_4 = 0.15,\ P_5 = 0.15$, respectively. Construct its Huffman code.

Codeword	x_i	P_i	
	x_1	0.25	0.3
	x_2	0.25	0.25
	x_3	0.2	0.25
0	x_4	0.15	0.2
1	x_5	0.15	

§ 3.4 Huffman Code

Codeword	x_i	P_i		
	x_1	0.25	0.3	0.45
0	x_2	0.25	0.25	0.3
1	x_3	0.2	0.25	0.25
0	x_4	0.15	0.2	
1	x_5	0.15		

Codeword	x_i	P_i			
1	x_1	0.25	0.3	0.45	0.55
0	x_2	0.25	0.25	0.3	0.45
1	x_3	0.2	0.25	0.25	
0 0	x_4	0.15	0.2		
0 1	x_5	0.15			

§ 3.4 Huffman Code

Codeword	x_i	P_i				
0 1	x_1	0.25	0.3	0.45	0.55	1
1 0	x_2	0.25	0.25	0.3	0.45	
1 1	x_3	0.2	0.25	0.25		
0 0 0	x_4	0.15	0.2			
0 0 1	x_5	0.15				

Validations:

$l_1 = 2, l_2 = 2, l_3 = 2, l_4 = 3, l_5 = 3$

$L = \sum_i l_i \cdot P_i = 2.3$ bits/symbol

$H_2(X) = \sum_i P_i \log_2 P_i^{-1} = 2.3$ bits/sym.

Q: Try to construct a Shannon-Fano code and see if it is also optimal.

§ 3.4 Huffman Code

So now, let us look back at the problem proposed at the beginning.
How to represent the source vector {1 2 4 4 3 1 4 4} ?

Codeword	x	$P(x)$			
0 1	1	0.25	0.25	0.5	1
0 0 0	2	0.125	0.25	0.5	
0 0 1	3	0.125	0.5		
1	4	0.5			

It should be represented as {0 1 0 0 0 1 1 0 0 1 0 1 1 1 } and $L = 1.75$ bits/symbol.

Q: How if the source vector becomes {1 2 4 3 4 4 2 1}?

Remark: The source coding depends on the source vector.

§ 3.4 Huffman Code

- Huffman code can also be defined as a *D*-ary code.

- A *D*-ary Huffman code can be similarly constructed following the binary construction.

Step 1: Merge the *D* smallest symbol probabilities;

Step 2: Assign the corresponding symbols with 0, 1, ..., $D - 1$, then go back to **Step 1**; Repeat the above process until *D* probabilities are merged into a probability of 1.

§ 3.4 Huffman Code

Example 3.5 Consider a source with symbols x_1, x_2, x_3, x_4, x_5, x_6. They occur with probabilities of $P_1 = 0.25$, $P_2 = 0.25$, $P_3 = 0.2$, $P_4 = 0.1$, $P_5 = 0.1$, $P_6 = 0.1$, respectively. Construct a ternary ({0, 1, 2}) Huffman code.

Codeword	x_i	P_i			
0	x_1	0.25	0.25	0.25	1
1	x_2	0.25	0.25	0.25	
2 0	x_3	0.2	0.2	0.5	
2 1	x_4	0.1	0.1		
2 2 0	x_5	0.1	0.2		
2 2 1	x_6	0.1			
2 2 2	Dummy	0			

Note: A dummy symbol is created such that 3 probabilities can merge into a probability of 1 in the end.

§ 3.4 Huffman Code

Properties on an optimal *D*-ary source code (Huffman code)

(1) If $p_j > p_k$, then $l_j \le l_k$;

(2) The D longest codewords have the same length;

(3) The D longest codewords differ only at the last symbol and correspond to the D least likely source symbols.

Theorem 3.5 (Optimal Source Code) A source code (C^*) is optimal if giving any other source code C', we have $L(C^*) \le L(C')$.

Note: Huffman codes are optimal.

References:

[1] Elements of Information Theory, by T. Cover and J. Thomas.

[2] Scriptum for the lectures, Applied Information Theory, by M. Bossert.