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 (in bits per symbol) implies in average the number of bits that is required to represent a source symbol. This indicates a mapping between source symbol and bits.
- Source coding can be seen as a mapping mechanism between symbols and 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 that be a shorter string of bits?

- **Definition:** Let x denote a source symbol and $C(x)$ denote a source codeword of x. If the length of $C(x)$ is $l(x)$ (in bits) and x happens with a probability of $p(x)$, then 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 is required to represent a 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 the following distributions:

$$X \in \{1,\ 2,\ 3,\ 4\}$$

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

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):

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

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

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

$\Rightarrow$ $L(C) > H(X)$.

Source Coding 2 (C^*):

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

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

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

$\Rightarrow$ $L(C^*) = H(X)$.

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

§ 3.1 An Introduction to Source Coding

Theorem 3.1 Shannon's Source Coding Theorem Given a memoryless source X whose symbols are chosen from the alphabet $\{x_1, x_2, \ldots, x_m\}$ with each alphabet symbol probabilities of $P(x_1) = p_1, P(x_2) = p_2, \ldots, P(x_m) = p_m$, and $\sum_{i=1}^{m} p_i = 1$. 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: $H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i$ bits/sym.

§ 3.1 An Introduction to Source Coding

Important features of source coding:

1. Unambiguous representation of source symbols (Non-singularity).

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

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

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

$$2\ 1 \begin{cases} \nearrow\ 3\ 2 \ldots \\ \searrow\ 4\ 2 \end{cases}$$

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 code

Definition: 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 instance you have reached a source codeword. The decoding is instantaneous, and the decoding output is '4 3 2 3 1 4 ' .

§ 3.2 Optimal Source Codes

- How can we find an optimal source code?
- An optimal source code :
  (1) Instantaneous code (prefix code)
  (2) Smallest expected length $L = \sum p_i l_i$

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 $l_1, l_2, \cdots, l_m$ must satisfy

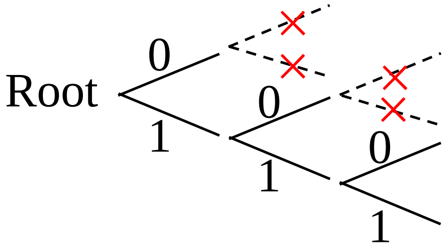
$$\sum_i D^{-l_i} \leq 1.$$

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

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

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

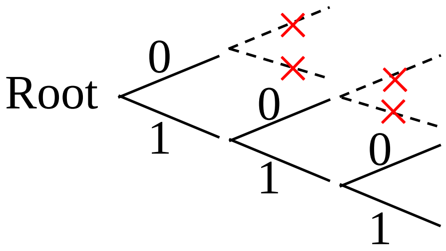
§ 3.2 Optimal Source Codes



Proof: - The above tree illustrates the assignment of source codeword symbols in a binary way when $D = 2$. A complete solid path represents a source codeword.

- Based on property of the instantaneous code, 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



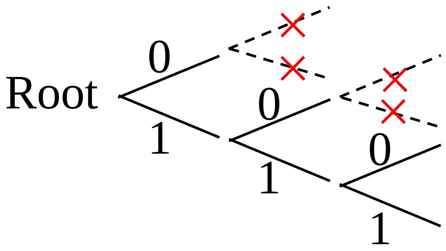
- At level $l_{\max}$ of the tree (source codeword length is $l_{\max}$), there are at most $D^{l_{\max}}$ codewords. Similarly, at level l_i of the tree, there are at most D^{l_i} codewords. All the codewords at level l_i have at most $D^{l_{\max}-l_i}$ descendants at level $l_{\max}$. Considering all levels l_i, the total number of descendants should not be greater than the maximal number of nodes at level $l_{\max}$ as

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

$$\downarrow$$

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

§ 3.2 Optimal Source Codes



- 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.

- The tree represents an instantaneous source code.

§ 3.2 Optimal Source Codes

- Finding the smallest expected length L becomes

  minimize: $L = \sum_i p_i l_i$

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

- The constrained minimization problem can be written as

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

  Lagrange Multipliers

- 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 $l_i^* = -\log_D p_i$.

§ 3.2 Optimal Source Codes

- With $l_i^* = -\log_D P_i$, we have

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

Entropy of the source symbols

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

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

Remark: since l_i can be 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

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

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

§ 3.2 Shannon-Fano Code

- Given a source that contains symbols $x_1, x_2, x_3, \ldots, x_m$ with probabilities of $p_1, p_2, p_3, \ldots, p_m$, 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 symbol x_i.

§ 3.2 Shannon-Fano Code

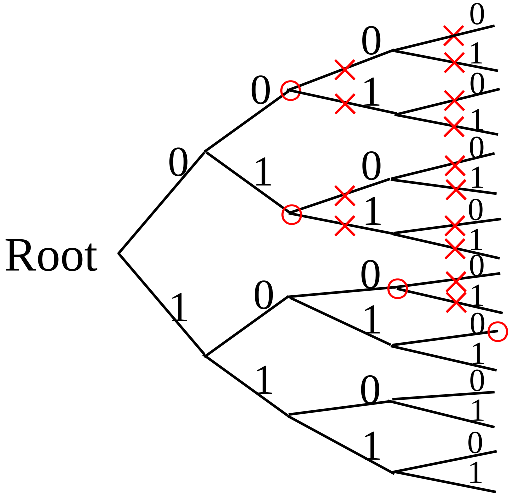
- **Example 3.3** Given a source with symbols x_1, x_2, x_3, x_4, they occur with probabilities of $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.2$, $p_4 = 0.1$, respectively. Construct its Shannon-Fano code.

We can determine

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

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.

§ 3.4 Huffman Code

- Given a source that contains symbols $x_1, x_2, x_3, \dots, x_m$ with probabilities of $p_1, p_2, p_3, \dots, p_m$, 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 Consider a random variable set of $X = \{1,\ 2,\ 3,\ 4,\ 5\}$. They have probabilities of $P(X=1) = 0.25,\ P(X=2) = 0.25, P(X=3) = 0.2,$ $P(X=4) = 0.15,\ P(X=5) = 0.15$. Construct a Huffman code to represent variable X.

Codeword	X	$P(X)$	
	1	0.25	0.3
	2	0.25	0.25
	3	0.2	0.25
0	4	0.15	0.2
1	5	0.15	

§ 3.4 Huffman Code

Codeword	X	$P(X)$		
	1	0.25	0.3	0.45
0	2	0.25	0.25	0.3
1	3	0.2	0.25	0.25
0	4	0.15	0.2	
1	5	0.15		

Codeword	X	$P(X)$			
1	1	0.25	0.3	0.45	0.55
0	2	0.25	0.25	0.3	0.45
1	3	0.2	0.25	0.25	
0 0	4	0.15	0.2		
0 1	5	0.15			

§ 3.4 Huffman Code

Codeword	X	$P(X)$				
0 1	1	0.25	0.3	0.45	0.55	1
1 0	2	0.25	0.25	0.3	0.45	
1 1	3	0.2	0.25	0.25		
0 0 0	4	0.15	0.2			
0 0 1	5	0.15				

Validations:

$l(1) = 2, l(2) = 2, l(3) = 2, l(4) = 3, l(5) = 3$

$L = \sum_X l(X) \cdot P(X) = 2.3$ bits/symbol

$H_2(X) = -\sum_X P(X) \log_2 P(X) = 2.3$ bits/symbol.

$$\boxed{L \geq H_2(X).}$$

§ 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.

Question: How if the source vector becomes $\{1\ 2\ 4\ 3\ 4\ 4\ 2\ 1\}$?

Remark: the Huffman code and its expected length depends on the source vector, i.e., entropy of the source.

§ 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 random variable set of $X = \{1, 2, 3, 4, 5, 6\}$. They have probabilities of $P(X = 1) = 0.25$, $P(X = 2) = 0.25$, $P(X = 3) = 0.2$, $P(X = 4) = 0.1$, $P(X = 5) = 0.1$, $P(X = 6) = 0.1$. Construct a ternary $\{0, 1, 2\}$ Huffman code.

Codeword	X	$P(X)$			
0	1	0.25	0.25	0.25	1
1	2	0.25	0.25	0.25	
2 0	3	0.2	0.2	0.5	
2 1	4	0.1	0.1		
2 2 0	5	0.1	0.2		
2 2 1	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 in 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 code is optimal.

References:

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

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