Coding for Discrete Sources
离散信源编码

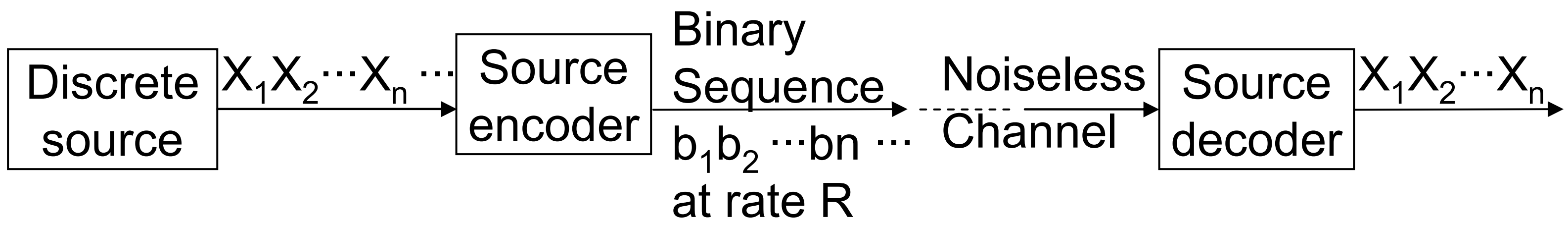


Fig 2.5 Lossless source coding

Definition: A lossless source code is a mapping from the set of source sequences into the set of binary sequences so that one can fully recover the original source sequences from the compressed binary sequences.

Lossless memoryless codes: A lossless memoryless code is a lossless source code which encodes source sequences symbol by symbol.

It is characterized by a mapping C from the source alphabet into the set of binary sequences:

$$\mathbf{x} \longrightarrow \mathbf{C(x)}$$

The output of the source encoder in response to the input $u_1u_2\cdots u_n$ is $C(u_1)C(u_2)\cdots C(u_n)$ (Symbols u_i's are encoded separately).

Lossless memoryless codes: Example

Symbol x	P(x)	$C_1(x)$	$C_2(x)$	$C_3(x)$
x_0	1/2	1	0	0
x_1	1/4	00	01	10
x_2	1/8	01	011	110
x_3	1/8	10	111	111

Suppose we are presented with the binary sequence 00100100

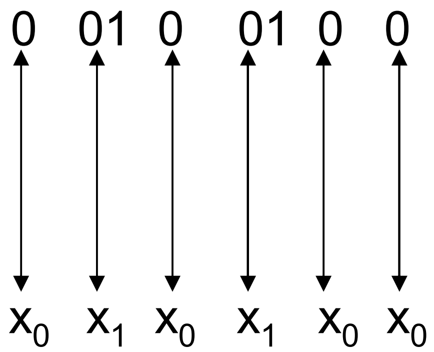
For C_1, $C_1(x_1)C_1(x_0)C_1(x_1)\ C_1(x_0)\ C_1(x_1) = 00100100$
$C_1(x_1)C_1(x_3)C_1(x_2)\ C_1(x_1) = 00100100$

The sequence 00100100 can be decoded by C_1 either as $x_1x_0x_1x_0x_1$ or as $x_1x_3x_2x_1$. Thus C_1 is not lossless.

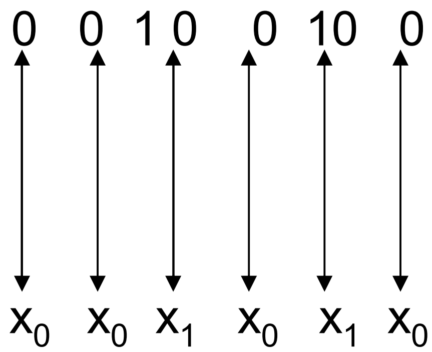
Lossless memoryless codes: Example (Continued)

Symbol x	P(x)	$C_1(x)$	$C_2(x)$	$C_3(x)$
x_0	1/2	1	0	0
x_1	1/4	00	01	10
x_2	1/8	01	011	110
x_3	1/8	10	111	111

For C_2



For C_3



One can check that C_2 and C_3 are a lossless memoryless code; the outputs of the corresponding source encoder in response to different input sequences are all different.

Lossless memoryless codes: Example (Continued)

- A lossless memoryless code is also called a uniquely decodable code. In this course, we consider only lossless codes. Memoryless codes always mean lossless, memoryless (or uniquely decodable codes).
- In both C_2 and C_3, 0 is a codeword corresponding to x_0. However, there is a striking difference between the decoding process of C_2 and C_3.
  - In the decoding by C_2 {0, 01, 011, 111}, one cannot decode 0 immediately as x_0. One has to wait to see several future digits before making a decision. For example, if the next digit is 1, then we cannot decode 0 as x_0.
  - On the other hand, in the decoding by C_3 (0, 10, 110, 111}, One can decode 0 immediately as x_0 without waiting to see future digits. A memoryless code having this kind of property is referred to as an instantaneous code.

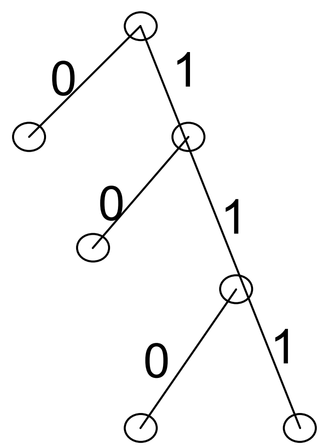
Lossless memoryless codes: Example (Continued)

• The difference between C_2 and C_3 lies in the fact that C_3 satisfies the so-called prefix property while C_2 does not. In C_3 each codeword is not a prefix of other codewords—this property is called the prefix property. 0 is not a prefix of 10, 110, and 111. Similarly, 10 is not a prefix of 110 and 111; 110 is not a prefix of 111. C3 can be represented by a binary tree.

• A memoryless code satisfying the prefix property is called a prefix code. C_3 is a prefix code. The prefix property and the instantaneously decodable property are the same.

• The construction of prefix code using tree:

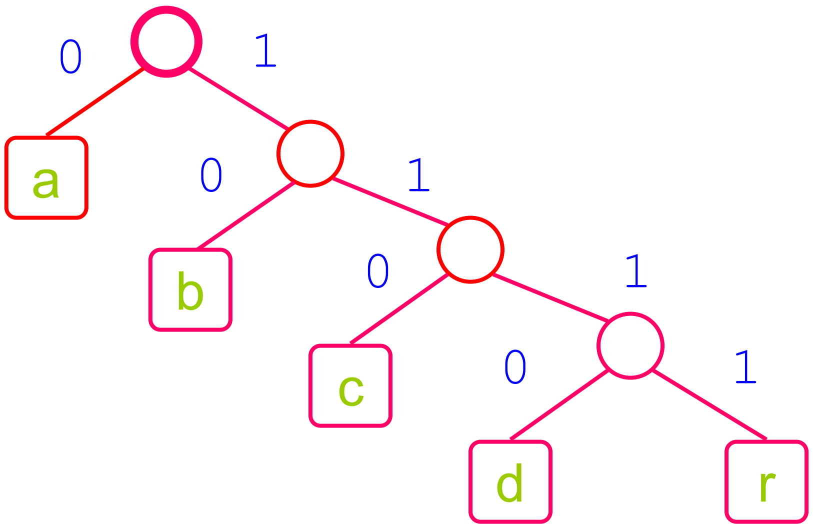
The code is from the terminal node. It guarantees that one code is not extended from another code.



The tree representation of C_3

Decoding of prefix code: Example

Symbol	code
a	0
b	10
c	110
d	1110
r	1111



0101111011001110010111110

= abracadabra

Performance of memoryless codes

- Let $\{X_i\}_{i=1}^{+\infty}$ be a discrete stationary source with a common marginal pmf p(x), x ε $\{x_0, x_1, \cdots, x_n\}$. Let C be a memoryless code. Let n_j be the length of the codeword $C(x_j)$. The performance of C is measured by its average codeword length in bits/symbol

$$\overline{R} = \sum_{j=0}^{J-1} p(x_j) n_j = E[\,|C(Xj)|\,],\ \forall j$$

|C(x)| = the length of C(x). $\overline{R}$ is the average rate of the output of C. In memoryless source coding, we look into how to construct a memoryless code C to minimize $\overline{R}$

Kraft-McMillan Inequality

Theorem 2.3.1:

- Any memoryless code C satisfies the following Kraft inequality

$$\sum 2^{-|c(x)|} = \sum_{j=0}^{J-1} 2^{-n_j} \leq 1$$

where n_j is the length of the codeword $C(x_j)$. (necessary condition)

- Given a set of codeword lengths n_j ($0 \leqslant j \leqslant J-1$) that satisfy the Kraft inequality, there exists a prefix code C such that $|C(x_j)|=n_j$ for any $0 \leqslant j \leqslant J-1$. (sufficient condition)

Proof of necessary condition

- For any m>0

$$\left[\sum_{j=0}^{J-1} 2^{-n_j}\right]^m = \left(\sum_{j_1=0}^{J-1} 2^{-n_{j_1}}\right)\left(\sum_{j_2=1}^{J-1} 2^{-n_{j2}}\right)\cdots\left(\sum_{j_m=1}^{J-1} 2^{-n_{jm}}\right)$$

$$= \sum_{j_1=1}^{J-1}\sum_{j_2=1}^{J-1}\cdots\sum_{j_m=1}^{J-1} 2^{-\left(n_{j_1}+n_{j2}+\ldots+n_{j_m}\right)}$$

- $n_{j_1}+n_{j_2}+\ldots+n_{j_m}$ is the code length for n symbols

Let $n_{\max} = \max\left(n_{j_1}, n_{j_2}, \ldots, n_{j_m}\right)$

$$m \le n_{j_1}+n_{j_2}+\ldots+n_{j_m} \le \mathrm{m} n_{\max}$$

Proof of necessary condition (Cntd)

$$\left[\sum_{j=0}^{J-1} 2^{-n_j}\right]^m = \sum_{k=m}^{mn_{\max}} A_k 2^{-k}$$

- A_k is the number of sequences such that $n_{j_1} + n_{j_2} + \ldots + n_{j_m} = k$
- Since C is lossless, and the number of binary sequences with length k is 2^k , it follows that $A_k \leqslant 2^k$. Thus

$$\left[\sum_{j=0}^{J-1} 2^{-n_j}\right]^m \le \sum_{k=m}^{mn_{\max}} 2^k 2^{-k} = mn_{\max} - m + 1$$

Therefore

$$\sum_{j=0}^{J-1} 2^{-n_j} \le (mn_{\max} - m + 1)^{1/m} \Rightarrow \sum_{j=0}^{J-1} 2^{-n_j} \le 1 \text{ when m} \to \infty$$

Proof of Sufficient condition (Cntd)

Assume $n_0 \leqslant n_1 \leqslant \cdots n_{J-1}$.

Since $$2^{-n_0} + 2^{-n_1} + \cdots 2^{-n_{J-1}} \leq 1$$

we have

$$2^{-n_{J-1}} \leq 1 - \sum_{j=0}^{J-2} 2^{-n_j}$$

Mutiply the two sides by $2^{n_{J-1}}$, we get

$$1 \leq 2^{n_{J-1}} - \sum_{j=0}^{J-2} 2^{-^{n_{J-1}} n_j}$$

Proof (Cntd)

From the given condition, we also have

$$\sum_{j=0}^{J-2} 2^{-n_j} \le 1$$

i.e., $2^{-n_{J-2}} \le 1 - \sum_{j=0}^{J-3} 2^{-n_j}$

Multiply the two sides by $2^{-n_{J-1}}$, we get

$$2^{n_{J-1}-n_{J-2}} \le 2^{n_{J-1}} - \sum_{j=0}^{J-3} 2^{n_{J-1}-n_j}$$

Continue,

$$2^{n_{J-1}-n_{J-3}} \le 2^{n_{J-1}} - \sum_{j=0}^{J-4} 2^{n_{J-1}-n_j}$$

$$\vdots$$

$$2^{n_{J-1}-n_1} \le 2^{n_{J-1}} - 2^{n_{J-1}-n_0}$$

In general, we have $2^{n_{J-1}-n_{j+1}} \le 2^{n_{J-1}} - \sum_{i=0}^{j-1} 2^{n_{J-1}-n_i}, \; j = 0,1,...,J-2$

Next we generate prefix codes by constructing a tree

a) Construct a full tree with n_{J-1} order.

b) Select a node with n_0 order as a terminal node, and assign a codeword accordingly. The total number of nodes is reduced by $2^{n_{J-1}-n_0}$ after removing nodes in the subtree of the node. The number of nodes remaining is $2^{n_{J-1}} - 2^{n_{J-1}-n_0}$

c) Next we select a node with n_1 order as a terminal node, and assign a codeword accordingly, the total number of nodes is reduced by $2^{n_{J-1}-n_1}$

The number of nodes remaining to be assigned is $2^{n_{J-1}} - \sum_{i=0}^{1} 2^{n_{J-1}-n_i}$

d) Continue this process till the nodes with n_{J-1} th order. The number of nodes left is $2^{n_{J-1}} - \sum_{i=0}^{J-2} 2^{n_{J-1}-n_i}$. Since $1 \le 2^{n_{J-1}} - \sum_{i=0}^{J-2} 2^{n_{J-1}-n_i}$, we can get a codeword with lengnth n_{J-1}.

This completes the construction of the prefix codes.

Example

X:	x_0	x_1	x_2	x_3
P(X)	½	¼	1/8	1/8

Code word length: $n_0=1$, $n_1=2$, $n_2=n_3=3$

$$\sum_{j=0}^{3} 2^{-n_j} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$$

As the codeword length satisfies the Kraft inequality,
we can find a prefix code.

Prefix code 1

x_0	$p(x_0)=1/2$	$C(x_0)=0$	$n_0=1$
x_1	$p(x_1)=1/4$	$C(x_1)=10$	$n_1=2$
x_2	$p(x_2)=1/8$	$C(x_2)=110$	$n_2=3$
x_3	$p(x_3)=1/8$	$C(x_3)=111$	$n_3=3$

Average codeword length is 14/8 bits/symbol

Prefix code 2

x_0	$p(x_0)=1/2$	$C(x_0)=111$	$n_0=3$
x_1	$p(x_1)=1/4$	$C(x_1)=110$	$n_1=3$
x_2	$p(x_2)=1/8$	$C(x_2)=10$	$n_2=2$
x_3	$p(x_3)=1/8$	$C(x_3)=0$	$n_3=1$

Average codeword length is 21/8 bits/symbol

Prefix code 1 is better as it makes use of source statistics.

Source Coding Theorem

Theorem 2.3.2

Let $\{X_i\}$ be a stationary source with a common marginal pmf p(x), $x \varepsilon \boldsymbol{X} = \{ x_0, x_1 \cdots, x_{J-1} \}$.

(a) The average codeword length $\overline{R}$ of any memoryless code C satisfies

$$\overline{R} \geq H(X_1)$$

(b) There is a prefix code C* such that

$$\overline{R}^* = \sum p(x) | C^*(x) | < H(X_1) + 1$$

Proof of Source Coding Theorem

(a) Let $n_j = |C(x_j)|$. Apply the log sum inequality, we get

$$\sum_{j=0}^{J-1} p(x_j)\log\frac{p(x_j)}{2^{-n_j}} \geq \sum_{j=0}^{J-1} p(x_j)\log\frac{\sum_{j=0}^{J-1} p(x_j)}{\sum_{j=0}^{J-1} 2^{-n_j}}$$

$$= -\log\sum_{j=0}^{J-1} 2^{-n_j} \geq 0$$

The last inequality is due to Kraft inequality.

$$\overline{R} = \sum_{j=0}^{J-1} p(x_j)n_j = \sum_{j=0}^{J-1} p(x_j)\log\frac{1}{2^{-n_j}}$$

$$\geq -\sum_{j=0}^{J-1} p(x_j)\log p(x_j) = H(X_1)$$

Proof of Source Coding Theorem (cntd)

(b) Assume that $p(x_j)>0$ for any $0\leqslant j \leqslant J-1$.

Let $n_j = \lceil -\log p,(x_j) \rceil \quad 0\leqslant j\leqslant J-1$.

where for any real number y, $\lceil y \rceil$ is equal to the least integer m such that $m \geqslant y$.

An important property of the function $\lceil y \rceil$ is $y \le \lceil y \rceil \le y+1$

Therefore $\quad -\log p(x_j) \le n_j \le -\log p(x_j) + 1$

We then have

$$\sum_{j=0}^{J-1} 2^{-n_j} \le \sum_{j=0}^{J-1} 2^{\log p(x_j)} = \sum_{j=0}^{J-1} p(x_j) = 1$$

From part (b) of theorem 2.3.1, it follows that there exists a prefix code C* such that $|C^*(x_j)|=n_j$. This implies that

$$\overline{R}^* = \sum_{j=0}^{J-1} p(x_j) n_j < \sum_{j=0}^{J-1} p(x_j)[-\log p(x_j)+1] = H(X_1)+1$$

Block Memoryless Codes

- There is at most 1 bit difference between the average length of the best momoryless code and the entropy $H(x_1)$.
- A block memoryless code can be used to reduce this difference. It is a lossless source code which encodes source sequences block by block. The block length is n.
- It can be described by a mapping C from the extended alphabet $X^n=\{ u_1, u_2, \cdots, u_n: u_i \varepsilon \boldsymbol{X}, 1 \leqslant i \leqslant n\}$ to the set of binary sequences:

$$u_1, u_2, \cdots, u_n \longrightarrow C\,(u_1, u_2, \cdots, u_n)$$

- All definitions and properties of memoryless codes can be carried over to the case of block memoryless codes.

Block Memoryless Coding Theorem

Theorem 2.3.3

Let $\{X_i\}$ be a stationary source with a common pmf p(x), x ε ***X.***

(a) For any block memoryless code C with block length n

$$\frac{1}{n}\overline{R} = \frac{1}{n}\sum_{u_1\cdots u_n \in X^n} p(u_1\cdots u_n)\,|C(u_1\cdots u_n)| \geq \frac{1}{n}H(x_1\cdots x_n)$$

(b) There is a prefix code C* with block length n such that

$$\frac{1}{n}\overline{R} = \frac{1}{n}\sum_{u_1\cdots u_n \in X^n} p(u_1\cdots u_n)\,|C^*(u_1\cdots u_n)| \leq \frac{1}{n}H(x_1\cdots x_n) + \frac{1}{n}$$

(c) The ultimate compression rate in bits/symbol of $\{X_i\}$ is given by

$$H_\infty(X) = \lim_{n\to\infty}\frac{1}{n}H(x_1\cdots x_n)$$

Proof of Theorem 2.3.3 c)

$H(X_1, \cdots, X_n)=H(X_1)+H(X_2|X_1)+\cdots+H(X_n|X_1 \cdots, X_{n-1})$

Because $H(X_n|X_1 \cdots, X_{n-1}) \leqslant H(X_n|X_2 \cdots, X_{n-1})= H(X_{n-1}|X_1 \cdots, X_{n-2})$

We have $\{H(X_n | X_1 \cdots X_{n-1})\}_{n=1}^{\infty}$ is nonincreasing and nonnegative.

Therefore $\lim_{n\to\infty} H(X_n | X_1 \cdots X_{n-1})$ exists

$$\lim_{n\to\infty} \frac{1}{n} H(X_n | X_1 \cdots X_{n-1}) = \lim_{n\to\infty} \frac{1}{n} [H(X_1) + \cdots + H(X_n | X_1 \cdots X_{n-1})]$$
$$= \lim_{n\to\infty} H(X_n | X_1 \cdots X_{n-1})$$

The quantity $H\infty(X)$ is called the entropy rate of the stationary source $\{X_i\}$.

It represents the content of the source {Xi} in terms of bits/symbol.

Huffman Coding

- Huffman coding is used to encode or compress data such as fax, ASCII text.
- It is proposed by Dr. David A. Huffman in 1952

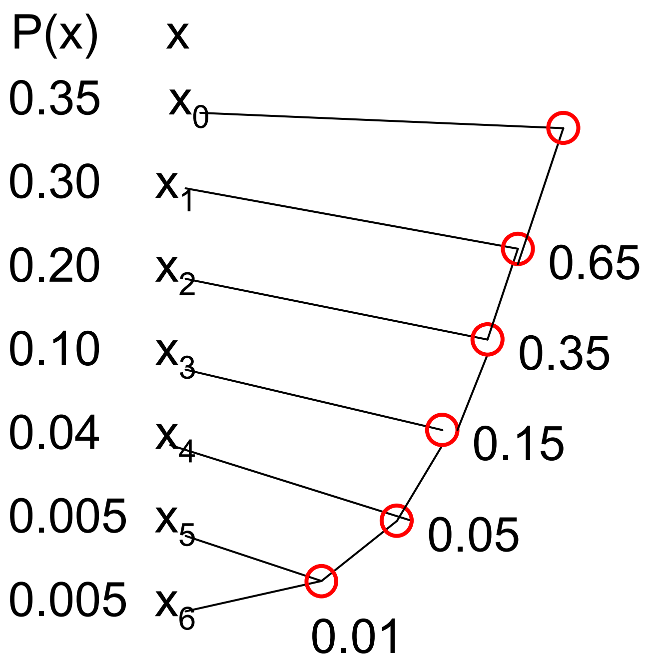
  "*A Method for the Construction of Minimum Redundancy Codes*"

- Huffman coding is a form of statistical coding, an optimal memoryless code C such that the average codeword length of C is minimized.
- Code word lengths vary and will be shorter for the more frequently used characters.

Huffman coding algorithm

a) Merge two symbols with the least probabilities into a symbol whose probability is equal to the sum of the two least probabilities.

b) Repeat a) until one symbol is left.

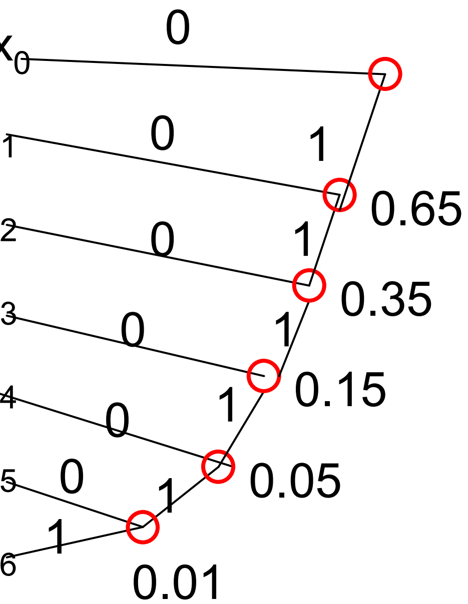


We get a binary tree in which the terminal nodes represent symbols in the original source alphabet and all other nodes represent merged symbols.

Huffman coding algorithm (Cntd)

c) Label two braches (leaves) emanating from each non-terminal node as 0 and 1. The code codeword of x_j is the binary sequence read from the root to the terminal node corresponding to x_j

P(x)	x	C(X)
0.35	x_0	0
0.30	x_1	10
0.20	x_2	110
0.10	x_3	1110
0.04	x_4	11110
0.005	x_5	111110
0.005	x_6	111111



C is a prefix code. H(X)=2.11 and $\overline{R}$ =2.21

Huffman coding algorithm (Cntd)

Examples 2 and 3 in pages 55 and 56

1) The code given by the Huffman coding algorithm is a prefix code and has the minimum average codeword length.
2) The Huffman encoding process is not unique. Different methods for labeling branches and different choices of merging symbols will give rise to different prefix codes.