Basic Concepts of Encoding

Redundancy vs. Efficiency

- 100% efficiency of encoding means that the average word length must be equal to the entropy of the original message ensemble:

$$\text{Efficiency} = \frac{H(X)}{\bar{L} \cdot \log D} \cdot 100\% \underset{\text{for} D=2}{=} \frac{H(X)}{\bar{L}}$$

if $D=2$ then $\log D = 1$

- If the entropy of the original message ensemble is less than the length of the word over the original alphabet, this means that the original encoding is redundant and that the original information may be compressed by the efficient encoding.

Redundancy vs. Efficiency

- On the other hand, as we have seen, to be able to detect and to correct the errors, a code must be redundant, that is its efficiency must be lower that 100%: the average word length must be larger than the entropy of the original message ensemble:

$$\text{Efficiency} = \frac{H(X)}{\bar{L} \cdot \log D} \cdot 100\% \underset{\text{for} D=2}{=} \frac{H(X)}{\bar{L}}$$

$$\text{if } D=2 \text{ then } \log D = 1$$

Redundancy and Error Correction

- As we have seen, the capacity of the channel is the maximum of transinformation (with respect to all possible sets of probabilities that could be assigned to the source alphabet) that could be transmitted through this channel:

$$C = \max I(X;Y) = \max\left[H(X) - H(X \mid Y)\right] =$$
$$= \max\left[H(Y) - H(Y \mid X)\right]$$

Redundancy and Error Correction

- For the digital communication channel with the binary alphabet, the probability of error (inversion of a bit) p and the probability of the correct transmission 1-p:

$$\max H(X) = 1;$$

$$H(X \mid Y) = H\left(P_{P=\begin{cases} p \\ 1-p \end{cases}} \right) = -\left(p \log p + (1-p) \log(1-p) \right)$$

$$C = H(X) - H(X \mid Y) = 1 + p \log p + (1-p) \log(1-p)$$

Redundancy and Error Correction

- The capacity C determines the limit for error correction encoding: if we need to transmit a message of the length m (bits) ensuring the error correction ability, we will need to transmit at least $n \geq m / C$ bits.

Redundancy and Error Correction

- Theorem. Let us have a digital communication channel with the probability of error p. Any error correction encoding, which ensures that the probability of the error in the transmitted word does not exceed ε, leads to $k_\varepsilon(m, p)$ times extension of the original information and

$$\lim_{\substack{\varepsilon \to 0 \\ m \to \infty}} k_\varepsilon(m, p) \geq 1/C$$

Redundancy and Error Correction

- The efficient encoding is reached when

$$\lim_{\substack{\varepsilon \to 0 \\ m \to \infty}} k_{\varepsilon}\left(m, p\right) = 1 / C$$

- The absolutely reliable encoding procedure does not exist because

$$k_0\left(m, p\right) = \infty$$

Shannon-Fano Encoding

- Sources without memory are such sources of information, where the probability of the next transmitted symbol (message) does not depend on the probability of the previous transmitted symbol (message).
- Separable codes are those codes for which the unique decipherability holds.
- Shannon-Fano encoding constructs reasonably efficient separable binary codes for sources without memory.

Shannon-Fano Encoding

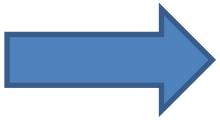
- Shannon-Fano encoding is the first established and widely used encoding method. This method and the corresponding code were invented simultaneously and independently of each other by C. Shannon and R. Fano in 1948.

Shannon-Fano Encoding

- Let us have the ensemble of the original messages to be transmitted with their corresponding probabilities:

$$[X]=[x_1, x_2, \ldots, x_n]; [P]=[p_1, p_2, \ldots, p_n]$$

- Our task is to associate a sequence C_k of binary numbers of unspecified length n_k to each message x_k such that:



Shannon-Fano Encoding

- No sequences of employed binary numbers C_k can be obtained from each other by adding more binary digits to the shorter sequence (prefix property).
- The transmission of the encoded message is "reasonably" efficient, that is, 1 and 0 appear independently and with "almost" equal probabilities. This ensures transmission of "almost" 1 bit of information per digit of the encoded messages.

Shannon-Fano Encoding

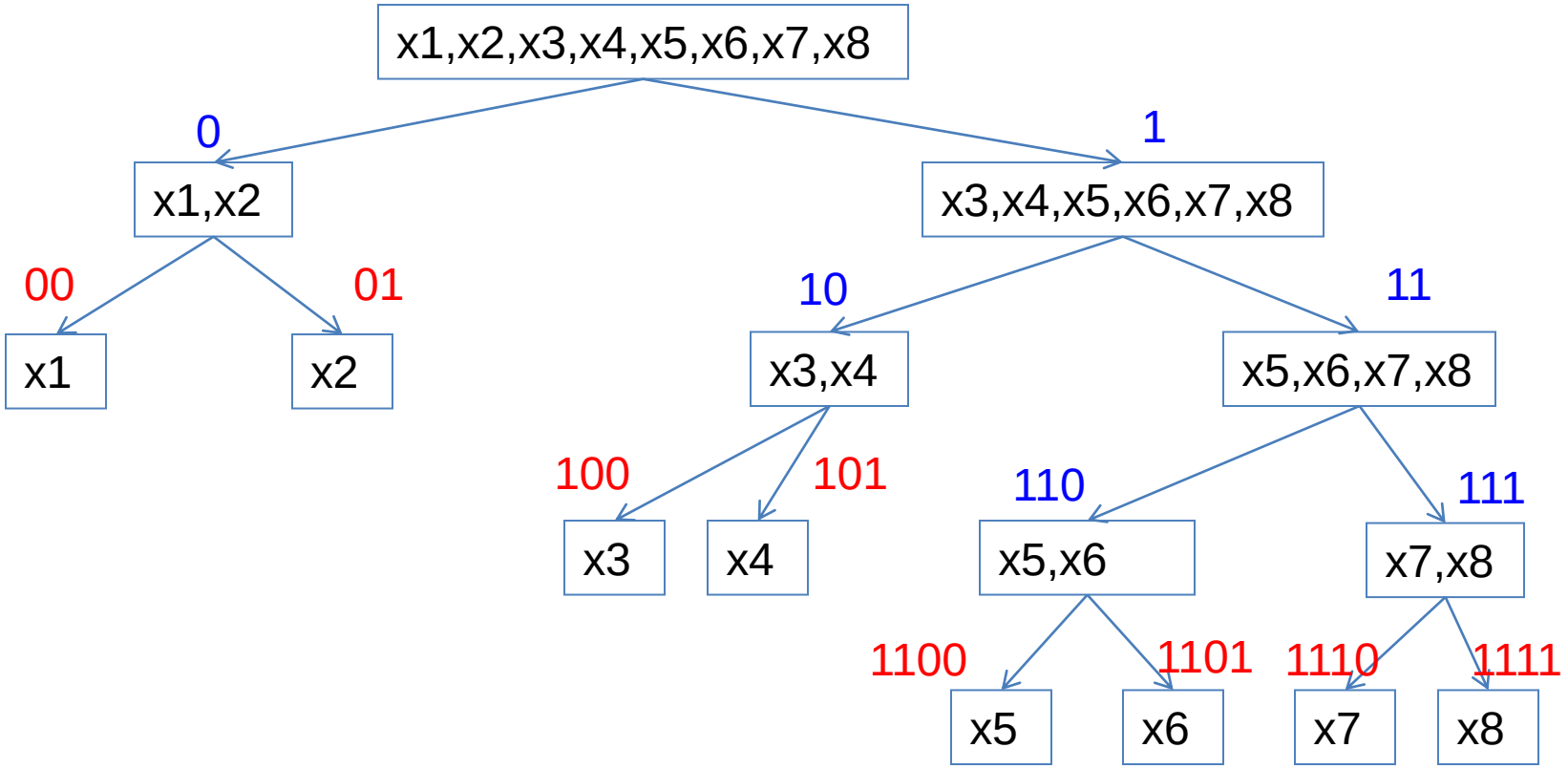
- Another important general consideration, which was taken into account by C. Shannon and R. Fano, is that (as we have already considered) a more frequent message has to be encoded by a shorter encoding vector (word) and a less frequent message has to be encoded by a longer encoding vector (word).

Shannon-Fano Encoding: Algorithm

- The letters (messages) of (over) the input alphabet must be arranged in order from most probable to least probable.
- Then the initial set of messages must be divided into two subsets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1".
- The same process is repeated on those subsets, to determine successive digits of their codes, as long as any sets with more than one member remain.
- When a subset has been reduced to one symbol, this means the symbol's code is complete.

Shannon-Fano Encoding: Example

Message	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Probability	0.25	0.25	0.125	0.125	0.0625	0.0625	0.0625	0.0625



Shannon-Fano Encoding: Example

Message	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Probability	0.25	0.25	0.125	0.125	0.0625	0.0625	0.0625	0.0625
Encoding vector	00	01	100	101	1100	1101	1110	1111

- Entropy $H = -\left(2 \cdot \left(\frac{1}{4}\log\frac{1}{4}\right) + 2 \cdot \left(\frac{1}{8}\log\frac{1}{8}\right) + 4 \cdot \left(\frac{1}{16}\log\frac{1}{16}\right)\right) = 2.75$
- Average length of the encoding vector

$$\bar{L} = \sum P\{x_i\} n_i = \left(2 \cdot \left(\frac{1}{4} \cdot 2\right) + 2 \cdot \left(\frac{1}{8} \cdot 3\right) + 4 \cdot \left(\frac{1}{16} \cdot 4\right)\right) = 2.75$$

- The Shannon-Fano code gives 100% efficiency

Shannon-Fano Encoding: Example

Message	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Probability	0.25	0.25	0.125	0.125	0.0625	0.0625	0.0625	0.0625
Encoding vector	00	01	100	101	1100	1101	1110	1111

- The Shannon-Fano code gives 100% efficiency. Since the average length of the encoding vector for this code is 2.75 bits, it gives the 0.25 bits/symbol compression, while the direct uniform binary encoding (3 bits/symbol) is redundant.

Shannon-Fano Encoding: Properties

- The Shannon-Fano encoding is the most efficient when the probability of the occurrence of each message (letter) x_k is of the form $P\{x_k\} = 2^{-n_k}$ and $\sum_{k=1}^{N} 2^{-n_k} = 2^{-n_1} + 2^{-n_2} + \ldots + 2^{-n_N} = 1$
- The prefix property always holds and

$$\bar{L} = \sum_{k=1}^{N} P\{x_k\} n_k = \underbrace{-\sum_{k=1}^{N} P\{x_k\} \underbrace{\log P\{x_k\}}_{-I(x_k)}}_{H(X)}$$

the efficiency is 100%.

Shannon-Fano Encoding: Properties

- It should be taken into account that the Shannon-Fano code is not unique because it depends on the partitioning of the input set of messages, which, in turn, is not unique.
- If the successive equiprobable partitioning is not possible at all, the Shannon-Fano code may not be an optimum code, that is, a code that leads to the lowest possible average length of the encoding vector for a given D.

Huffman Encoding

- This encoding algorithm has been proposed by David A. Huffman in 1952, and it is still the main loss-less compression basic encoding algorithm.
- The Huffman encoding **ensures** constructing separable codes (the unique decipherability property holds) with **minimum redundancy** for a set of discrete messages (letters), that is, this encoding results in an optimum code.

Huffman Encoding: Background

- For an optimum encoding, the longer encoding vector (word) should correspond to a message (letter) with lower probability:

$$P\{x_1\} \geq P\{x_2\} \geq \ldots \geq P\{x_N\} \leftrightarrow L\{x_1\} \leq L\{x_2\} \leq \ldots \leq L\{x_N\}$$

- For an optimum encoding it is necessary that

$$L(x_{N-1}) = L(x_N)$$

otherwise the average length of the encoding vector will be unnecessarily increased.

- It is important to mention that not more than *D* (*D* is the number of letters in the encoding alphabet) encoding vectors could have equal length (for the binary encoding *D*=2)

Huffman Encoding: Background

- For an optimum encoding with D=2 it is necessary that the last two encoding vectors are identical except for the last digits.
- For an optimum encoding it is necessary that each sequence of length $L(x_N)-1$ digits either must be used as an encoding vector or must have one of its prefixes used as an encoding vector.

Huffman Encoding: Algorithm

- The letters (messages) of (over) the input alphabet must be arranged in order from most probable to least probable.
- Two least probable messages (the last two messages) are merged into the composite message with a probability equal to the sum of their probabilities. This new message must be inserted into the sequence of the original messages instead of its “parents”, accordingly with its probability.
- The previous step must be repeated until the last remaining two messages will compose a message, which will be the only member of the messages’ sequence.
- The process may be utilized by constructing a binary tree – the Huffman tree.

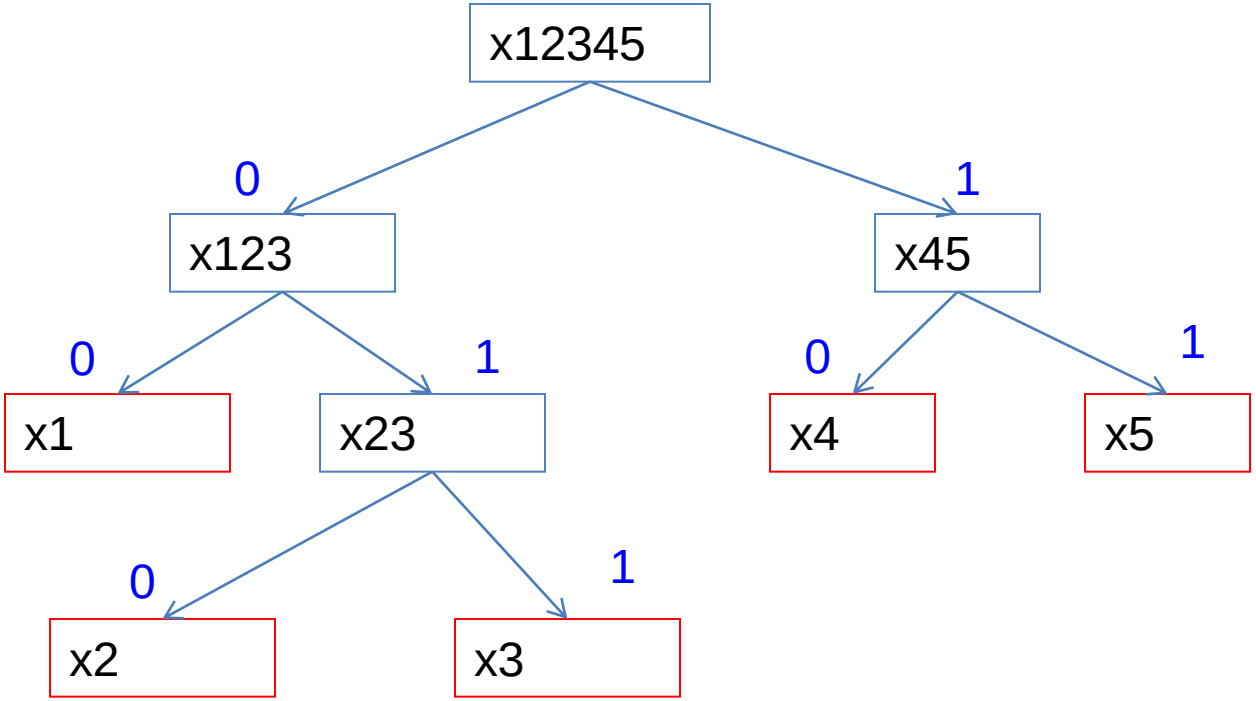
Huffman Encoding: Algorithm

- The Huffman tree should be constructed as follows: **1**) A root of the tree is a message from the last step with the probability 1; **2**) Its children are two messages that have composed the last message; **3**) The step 2 must be repeated until all leafs of the tree will be obtained. These leafs are the original messages.
- The siblings-nodes from the same level are given the numbers 0 (left) and 1 (right).
- The encoding vector for each message is obtained by passing a path from the root's child to the leave corresponding to this message and reading the numbers of nodes (root's child→intermidiates→leaf) that compose the encoding vector.

Huffman Encoding: Example

- Let us construct the Huffman code for the following set of messages: x1, x2, x3, x4, x5 with the probabilities p(x1)=…=p(x5)=0.2
- 1) x1 (p=0.2), x2 (p=0.2), x3 (p=0.2), x4 (p=0.2), x5 (p=0.2)
- 2) x4,x5→x45 (p=0.4)=> x45,x1,x2,x3
- 3) x2,x3→x23 (p=0.4)=>x45, x23, x1
- 4) x1,x23→x123(p=0.6)=> x123, x45
- 5) x123, 45→x12345 (p=1)

Huffman Encoding: Example



Encoding vectors: x1→(00); x2→(010); x3→(011); x4→(10); x5→(11)

Huffman Encoding: Example

- Entropy $H(X) = -5(0.2\log 0.2) = -5\left(\frac{1}{5}\log\frac{1}{5}\right) = -\log\frac{1}{5} = \log 5 \approx 2.32$
- Average length of the encoding vector

$$\bar{L} = 3\cdot\left(\frac{1}{5}\cdot 2\right) + 2\left(\frac{1}{5}\cdot 3\right) = \frac{12}{5} = 2.4$$

- The Huffman code gives (2.32/2.4)100% = 97% efficiency