Information Theory and Channel Coding

Prof. Rodrigo C. de Lamare
CETUC, DEE, PUC-Rio, Brazil
delamare@puc-rio.br

II. Source coding

- Source coding corresponds to the compression of data and information theory describes the ultimate limits of data compression.

- Shannon established this fundamental limit which corresponds to the entropy of the source in 1948 through the source coding theorem.

- We first consider lossless source coding techniques that do not lead to any loss of information.

- Then we examine lossy compression approaches that are often used in multimedia but which imply loss of information.

This chapter deals with source coding techniques and is structured as:

A. Fundamentals

B. Source coding theorem

C. Prefix coding

D. Huffman coding

E. Lempel-ziv coding

F. Quantisation

A. Fundamentals

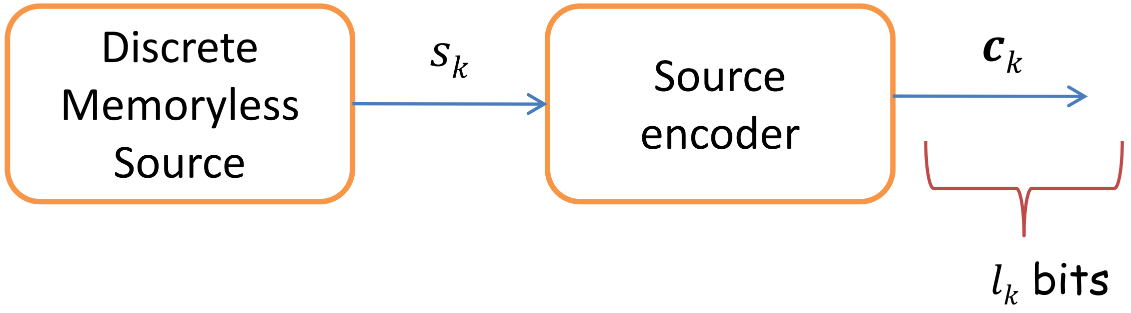
- Source coding is the process of representing data generated by a discrete source in an efficient manner.

- In this context, the knowledge of the statistics of the source can help the encoding and increase the efficiency.

- In our exposition, we will assume the following:

  - Use of binary codewords.
  - The source code is uniquely decodable.
  - The source has an alphabet with K symbols, i.e., $\xi = \{s_0, s_1, \dots, s_{K-1}\}$.
  - The kth symbol s_k occurs with probability p_k, $k = 0,1,\dots,K-1$.
  - The binary codeword $\boldsymbol{c}_k$ assigned to symbol s_k has length l_k in bits.

- A general source coding scheme is illustrated as follows:



- The average codeword length is given by

$$\bar{l} = \sum_{k=0}^{K-1} p_k l_k \quad \text{bits}$$

where the above corresponds to the average number of bits used to encode the source symbols.

- Efficiency of source coding:

$$\eta = \frac{l_{\min}}{\bar{l}},$$

where l_{min} is the smallest possible average number of bits of the codeword, also known as average codeword length.

- How do we obtain l_{min}?

→ The first theorem of Shannon: “The source coding theorem”

B. The source coding theorem

- Given a discrete memoryless source with entropy $H(\xi)$, the average codeword length for any lossless encoding scheme is bounded by

$$\bar{l} \geq H(\xi)$$

- The entropy H(ξ) is the fundamental limit of compression, i.e., the limit to the average number of bits per source symbol required to represent a discrete memoryless source.

- In a source encoding scheme, when $l_{min} = H(\xi)$, the efficiency is given by

$$\eta = \frac{H(\xi)}{\bar{l}}$$

Shannon, Claude Elwood (July 1948). "A Mathematical Theory of Communication" (PDF). *Bell System Technical Journal*. **27** (3): 379–423.

Example 1

Consider the following symbols and probabilities associated with a discrete memoryless source and the codes employed.

Source symbols	Probabilities	Code
s_0	0.5	0
s_1	0.25	10
s_2	0.15	110
s_3	0.1	111

a) Compute the entropy of the source

b) Calculate the average codeword length and the efficiency of the codes

Solution:

a) $H(\xi) = \sum_{k=0}^{K-1} p_k log_2\left(\frac{1}{p_k}\right) = 0.5 \times 1 + 0.25 \times 2 + 0.15 \times log_2\left(\frac{1}{0.15}\right) + 0.1 \times log_2(10) = 1.7427$ bits

b) $\bar{l} = \sum_{k=0}^{K-1} p_k l_k = 0.5 \times 1 + 0.25 \times 2 + 0.15 \times 3 + 0.1 \times 3 = 1.75$ bits

$$\eta = \frac{H(\xi)}{\bar{l}} = 99.59\ \%$$

C. Prefix coding

- Since sources often exhibit some form of redundancy, it is possible to increase the transmission efficiency through data compression.

- Data compression could be of two forms:

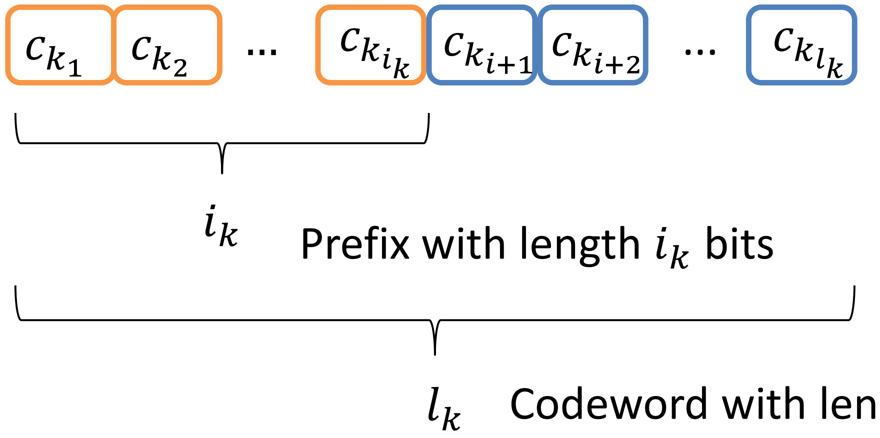
    - Lossless -> with no loss of information

    - Lossy -> with loss of information

- Prefix coding can obtain an average codeword length $\bar{l}$ that could become arbitrarily close to the entropy $H(\xi)$.

- Let us consider a discrete memoryless source with alphabet $\xi = \{s_0, s_1, \dots, s_{K-1}\}$ with probabilities $\{p_0, p_1, \dots, p_{K-1}\}$.

- We assume that the codewords are uniquely decodable and the prefix condition



- Any sequence that contains the initial part of the codeword is a prefix.

Example 2

Consider the following symbols and probabilities associated with a discrete memoryless source and the codes employed.

Source symbols	Probabilities	Code A	Code B	Code C
s_0	0.5	0	0	0
s_1	0.25	1	10	01
s_2	0.15	00	110	011
s_3	0.1	11	111	0111

Analyze the codes and determine if they are prefix codes.

Solution:

Code A is not a prefix code since the bit 0, the codeword for s_0, is a prefix of 00, the codeword for s_2. Likewise, the codeword for s_1, the bit 1, is a prefix of 11, the codeword for s_3.

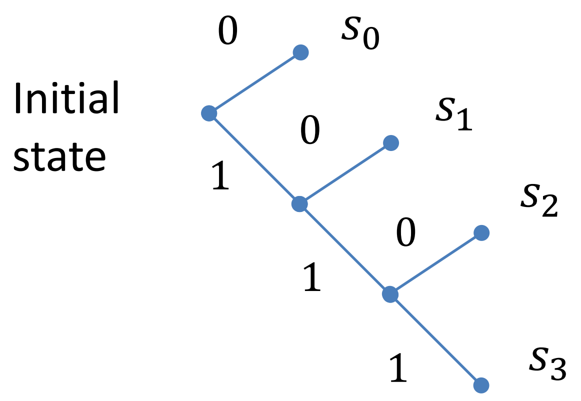
For similar reasons, code C is also not a prefix code.

Code B is a prefix code as all the prefixes of the codewords are unique.

Decoding of prefix codes

- The decoder of prefix codes inspects the beginning of a sequence and decodes one codeword at each time instant.

- Specifically, we employ a decision tree for code B described by



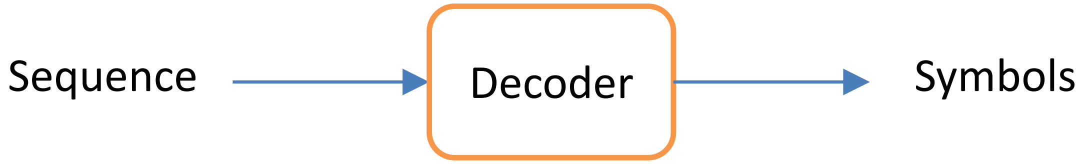
Example 3

Consider the following symbols and codes produced by a discrete memoryless source.

Source symbol	Code
s_0	0
s_1	10
s_2	110
s_3	111

Describe in detail the decoding of the sequence $s = \{1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\}$

Solution:



The sequence $s = \{1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\}$ produces the sequence of symbols given by

$$s_1 s_3 s_2 s_0 s_0$$

The decoding can be performed by inspecting the sequence of bits and matching to the codewords in the table.

Properties

i) Uniquely decodable

ii) Kraft-McMillan inequality

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

Assuming binary codewords, the lengths of the codewords must always satisfy the above inequality.

Kraft-McMillan inequality

- Let us consider a discrete memoryless source with alphabet $\xi = \{s_0, s_1, \dots, s_{K-1}\}$ with probabilities $\{p_0, p_1, \dots, p_{K-1}\}$.

- Let us also assume that we have K binary codewords $\boldsymbol{c}_k$, $k = 0,1, \dots, K-1$ with lengths $\{l_0, l_1, \dots, l_{K-1}\}$.

- The codeword lengths must satisfy the Kraft-McMillan inequality

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

- The inequality shows that one can construct a prefix code $\boldsymbol{c}_k$, $k = 0,1, \dots, K-1$, with lengths $-\log_2 p_k$.

Proof

Let us consider a prefix code in a tree (remember the decoding of prefix codes) and let

$$l_{max} = \max\{l_0, \dots, l_{K-1}\}$$

By expanding the tree such that all branches have depth l_{max}, we obtain a codeword with depth l_k with $2^{l_{max}-l_i}$ branches.

Since the sets of branches associated with codewords are disjoint, the total number of branches associated with codewords is less than $2^{l_{max}}$.

Therefore, we have

$$\sum_{k=0}^{K-1} 2^{l_{max}-l_k} \le 2^{l_{max}}$$

By manipulating the terms, we obtain the Kraft-McMillan inequality

$$2^{l_{max}} \sum_{k=0}^{K-1} 2^{-l_k} \leq 2^{l_{max}}$$

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

Implications of the Kraft-McMillan inequality

- The average codeword length $\bar{l}$ is bounded by

$$H(\xi) \leq \bar{l} < H(\xi) + 1$$

- The lower bound is satisfied with equality if c_k is produced by the source with probability

$$p_k = 2^{-l_k},$$

where l_k is the length of the designated codeword. This leads to optimal codes.

- Therefore, we have

$$\sum_{k=0}^{K-1} 2^{-l_k} = \sum_{k=0}^{K-1} p_k = 1$$

Optimal prefix codes

- By chosing codes with a specific relation between their probabilities and lengths, we can obtain optimal prefix codes that yield

$$\bar{l} = \sum_{k=0}^{K-1} p_k \, l_k \to H(\xi)$$

Proof

Let us consider the following optimization problem

$$\min \bar{l} = \sum_{k=0}^{K-1} p_k \, l_k$$

$$\text{subject to} \sum_{k=0}^{K-1} 2^{-l_k} \leq 1$$

We neglect at first the constraint on integers in l_k and suppose that the constraint is an equality.

We can then rewrite the optimization with constraints using the method of Lagrange multipliers and considering the Lagrangian

$$\mathcal{L} = \sum_{k=0}^{K-1} p_k \, l_k + \lambda \left(\sum_{k=0}^{K-1} 2^{-l_k} - 1 \right)$$

By differentiating the Lagrangian with respect to l_k, we obtain

$$\frac{\partial \mathcal{L}}{\partial l_k} = p_k - \lambda\, 2^{-l_k} \log_e 2$$

Equating the above to zero ($\frac{\partial \mathcal{L}}{\partial l_k} = 0$), we have

$$2^{-l_k} = \frac{p_k}{\lambda \log_e 2}$$

Substituting λ into the constraint, we get

$$\lambda = \frac{1}{\log_e 2}$$

Therefore, we obtain the optimal relation between probabilities and codeword lengths

$$p_k = 2^{-l_k} \text{ and } l_k = -\log_2 p_k$$

If we substitute the above relations into $\bar{l} = \sum_{k=0}^{K-1} p_k\, l_k$ then we obtain

$$\bar{l} = \sum_{k=0}^{K-1} p_k\, l_k = \sum_{k=0}^{K-1} p_k - \log_2 p_k$$

$$= -\sum_{k=0}^{K-1} p_k \log_2 p_k = H(\xi)$$

Further relations

- For optimal prefix codes, the Kraft-McMillan inequality also shows us that the average codeword length is given by

$$\bar{l} = \sum_{k=0}^{K-1} p_k \, l_k = \sum_{k=0}^{K-1} 2^{-l_k} \, l_k = \sum_{k=0}^{K-1} \frac{l_k}{2^{l_k}}$$

- The entropy of the source for $l_k = \log_2 \, 2^{l_k}$ is then given by

$$H(\xi) = \sum_{k=0}^{K-1} \frac{1}{2^{l_k}} \log_2 \, 2^{l_k} = \sum_{k=0}^{K-1} \frac{l_k}{2^{l_k}} = \bar{l}$$

- In this special case, we have $H(\xi) = \bar{l}$, which again verified the lower bound.

- The verification of the upper bound of $H(\xi) \leq \bar{l} < H(\xi) + 1$ can be done by examining how a prefix code can be matched to an arbitrary source.

- This can be done using an extended code.

- Let $\overline{l_n}$ be the average codeword length of a codeword associated with an extended codeword of n symbols, which results in

$$H(\xi^n) \leq \bar{l}_n < H(\xi^n) + 1$$

- Substituting the relation of entropy of an extended code into the above relation, we obtain

$$nH(\xi) \leq \bar{l}_n < nH(\xi) + 1$$

- By dividing the previous expression by n, we arrive at

$$H(\xi) \leq \frac{\bar{l}_n}{n} < H(\xi) + \frac{1}{n}$$

- If we take the limit when $n \to \infty$, we have

$$\lim_{n \to \infty} \frac{\bar{l}_n}{n} = H(\xi)$$

- This indicates that with n sufficiently large, we have

$$\bar{l} \to H(\xi)$$

- However, the above implies an increase in the computational complexity of decoding.

D. Huffman coding

- Basic ideas:
  - To assign to each symbol a code (sequence of bits) approximately equal in length to the amount of information in the symbol.
  - To substitute the set of statistics (probabilities) of the source by a second simpler set.
- The Huffman coding algorithm requires the statistics of the source, which can be obtained off-line, and approach the entropy of the source.
- It can be easily adapted to extended sources.

Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy Codes" (PDF). Proceedings of the IRE. ***40*** *(9): 1098–1101*

Huffman coding algorithm

i) Source symbols are listed in decreasing order of probability.

ii) The two symbols with lowest probabilities are designated 0 or 1.

iii) The two symbols above are combined into a new symbol with probability equal to the sum of the original probabilities.

iv) The new symbol is listed with the remaining symbols and their probabilities.

v) The procedure is repeated until only two symbols remain.

The code is the backward sequence of 0s and 1s obtained from the symbols.

The Huffman code is not unique but converges to the entropy $H(\xi)$

Example 4

Five symbols of the alphabet of a discrete memoryless source and their probabilities are shown below.

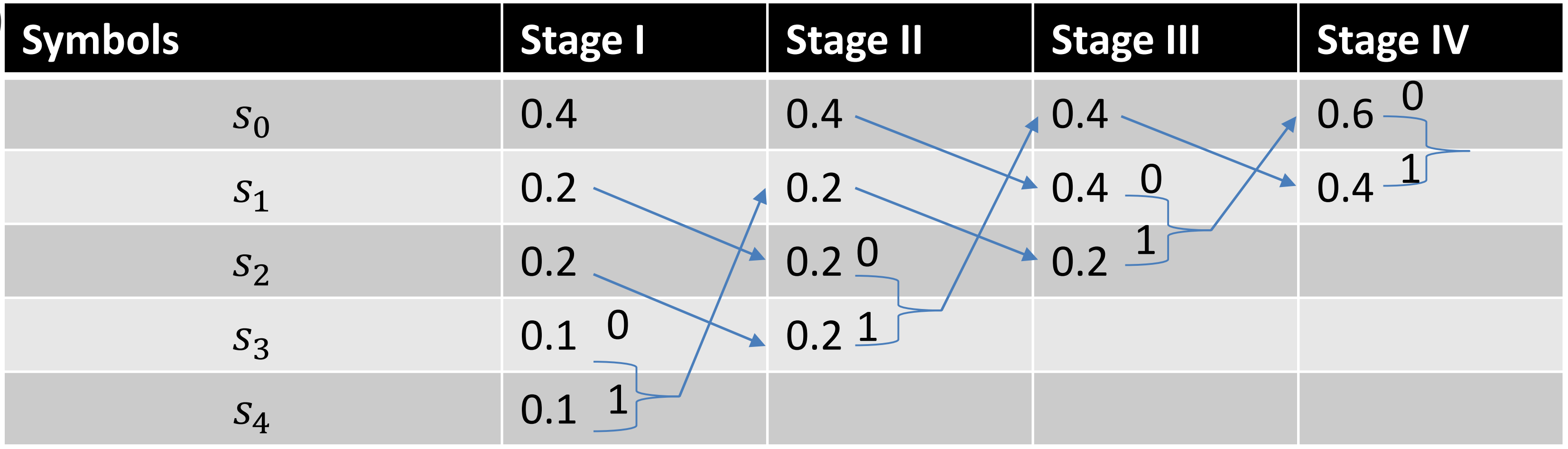
Source symbol	Probabilities
s_0	0.4
s_1	0.2
s_2	0.2
s_3	0.1
s_4	0.1

a) Perform the Huffman coding algorithm.

b) Compute the entropy, the average codeword length and the efficiency.

Solution:

a)



Symbols	Stage I	Stage II	Stage III	Stage IV
s_0	0.4	0.4	0.4	0.6 0
s_1	0.2	0.2	0.4 0	0.4 1
s_2	0.2	0.2 0	0.2 1	
s_3	0.1 0	0.2 1		
s_4	0.1 1			

Source symbol	Probabilities	Codes
s_0	0.4	00
s_1	0.2	10
s_2	0.2	11
s_3	0.1	010
s_4	0.1	011

b) The average codeword length is

$\bar{l} = \sum_{k=0}^{K-1} p_k \, l_k = 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 = 2.2$ bits

The entropy is given by

$$H(\xi) = \sum_{k=0}^{K-1} p_k log_2\left(\frac{1}{p_k}\right)$$
$$= 0.4 \ \log_2(1/0.4) + 0.2 \ \log_2(1/0.2) + 0.2 \ \log_2(1/0.2) + 0.1 \ \log_2(1/$$

E. Lempel-Ziv coding

- Invented by Lempel and Ziv in 1977 with extensions in 1978 and then later.
- It is a lossless universal compression scheme adopted for pdf, gif, zip and compress, which are widely used nowadays.
- Motivation:
  - Huffman coding requires knowledge of the statistics of the source.
  - Statistics might change with the source data to be compressed.
  - Need for a universal lossless compression approach that does not require the statistics of the source.

Ziv, J.; Lempel, A. (1978). "Compression of individual sequences via variable-rate coding" (PDF). IEEE Transactions on Information Theory. ***24*** *(5): 530.*

- Basic ideas:
  - To encode data by splitting or parsing them into blocks of symbols of variable length.
  - The blocks that are encoded have not been found previously.
  - A dictionary with codewords with l_k bits (fixed length) are used to encode the blocks.

Universal source compression

- We first set the benchmark using the performance of an optimal compressor that knows the source statistics.
- We construct a universal compression scheme that does not know the source statistics but is asymptotically optimal.
- Consider the problem of compressing a source sequence s^n with some source code.
- For the sake of brevity, we will consider the most common case that the source code outputs a binary sequence.
- The conclusions carry over to non-binary alphabets easily

- A source code for an n-block sequence $\boldsymbol{c_k}$ is defined as a mapping from a source sequence $\boldsymbol{s}^n$ to a binary sequence of finite length, i.e.,

$$\boldsymbol{c_k}\,(\boldsymbol{s}^n) = c_1 c_2 \ldots c_{l_k},$$

where l_k is the length of the output sequence and $c_i \in \{0,1\}$.

- For any source sequence and uniquely decodable code, we have

$$H(\boldsymbol{s}^n) \leq \bar{l} \leq H(\boldsymbol{s}^n) + 1$$

- Thus, when we consider a source random process s, and look at the average per-symbol description length, we have

$$\lim_{n \to \infty} E\left[\frac{1}{n}\bar{l}\,(\boldsymbol{s}^n)\right] = \lim_{n \to \infty} \frac{1}{n} H(\boldsymbol{s}^n) = H(\boldsymbol{s}),$$

where $H(\boldsymbol{s})$ is the entropy of the random process $\boldsymbol{s}$.

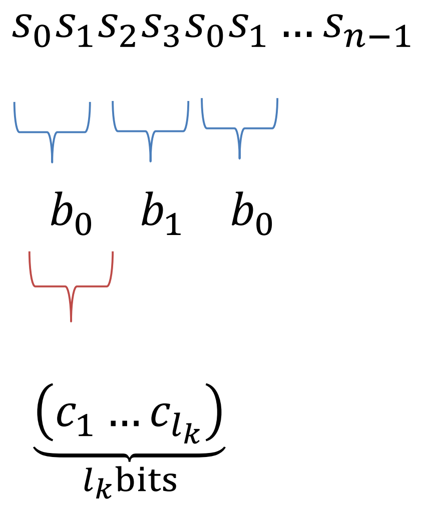
- An encoding scheme that produces sequences that lead to the above condition is universal.

- The Huffman code does not fall into this category due to their dependence on the source distribution.

- However, we will see one celebrated example of such a scheme: the Lempel-Ziv coding

Lempel-Ziv encoder

- The Lempel-Ziv encoder considers the following sequence of symbols:
- A sequence or block of symbols after parsing is obtained
- A dictionary employs binary codewords with l_k bits to encode the sequence of symbols.
- The Lempel-Ziv algorithm parses the sequence of symbols.
- The Lempel-Ziv code is the index of the dictionary that corresponds to l_k bits.



Lempel-Ziv decoder

- The decoding of the Lempel-Ziv codes requires knowledge of the initial dictionary and uses the following principles:
    - A pointer is employed to identify the codeword.
    - Once the codeword is identified, the original sequence of symbols is reconstructed.

$$\underbrace{\left(c_1 \dots c_{l_k}\right)}_{l_k\text{bits}} \quad \underbrace{\left(c_1 \dots c_{l_k}\right)}_{l_k\text{bits}} \dots \quad \underbrace{\left(c_1 \dots c_{l_k}\right)}_{l_k\text{bits}} \qquad \text{codewords}$$

$$\underbrace{\qquad\quad}_{b_i}$$

$$\underbrace{\qquad\quad}_{s_0 s_1} \qquad \text{symbols}$$

Example 5

Encode the following sequence using the Lempel-Ziv algorithm.

0100001 10000101000000101000001 1000000101000010

Solution:

Parsing the sequence by the rules previously explained results in the following blocks:

0, 1, 00, 001 , 10, 000, 101, 0000, 01, 010, 00001 , 100, 0001, 0100, 0010,

Clearly, all the blocks are different and each blocks is one of the previous blocks concatenated with a new source output.

The number of blocks is 15. This means that, for each block, we need 4 bits, plus an extra bit to represent the new source output.

The preceding sequence is encoded by

0000 0, 0000 1 , 0001 0, 0011 1 , 0010 0, 0011 0, 0101 1 , 0110 0, 0001 1, 1001 0, 1000 1, 0101 0, 0110 1, 1010 0, 0100 0

	Dictionary	Symbol	Codeword
1	0001	0	0000 0
2	0010	1	0000 1
3	0011	00	0001 0
4	0100	001	0011 1
5	0101	10	0010 0
6	0110	000	0011 0
7	0111	101	0101 1
8	1000	0000	0110 0
9	1001	01	0001 1
10	1010	010	1001 0
11	1011	00001	1000 1
12	1100	100	0101 0
13	1101	0001	0110 1
14	1110	0100	1010 0
15	1111	0010	0100 0

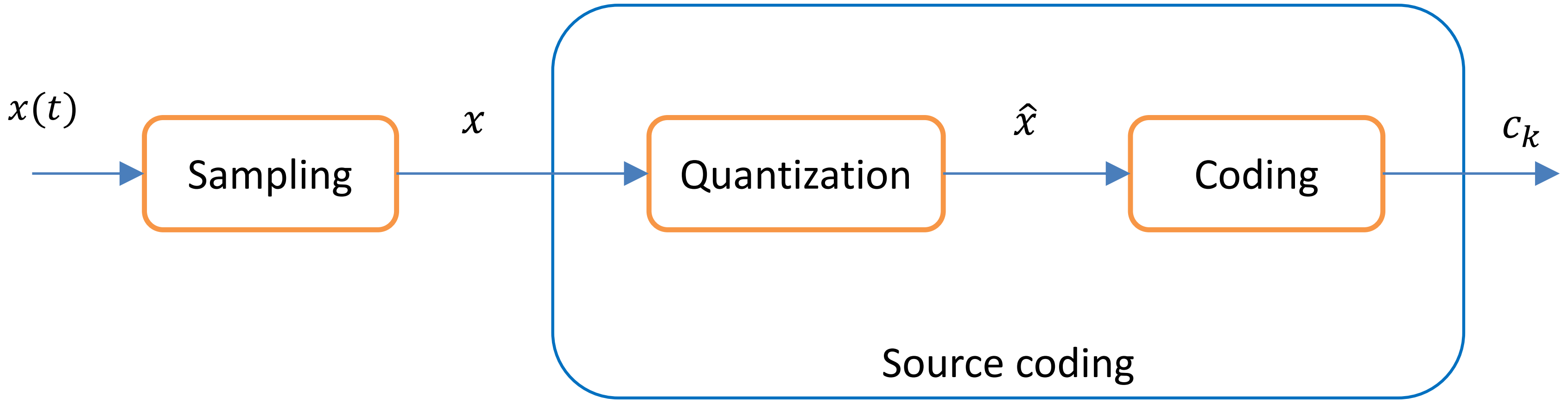
This representation can hardly be called a data compression scheme because a sequence of length 44 has been mapped into a sequence of length 75.

However, as the length of the original sequence is increased, the compression role of this algorithm becomes more apparent.

We can prove that for a stationary and ergodic source, as the length of the sequence increases, the number of bits in the compressed sequence approaches $H(s)$.

F. Quantization

- Given a bandlimited signal $x(t)$ obtained from a wide-sense stationary stochastic process, we can represent $x(t)$ using a sequence of samples.



- Quantization
  - Discretization of amplitudes of x
  - Minimization of a distortion
  - Lossy compression

- Simple encoding strategy:

$$\hat{x} \rightarrow \underbrace{\boldsymbol{c}_k}_{l_k \text{ bits}} \qquad \text{Binary codeword}$$

- Rate:

$$R = \log_2 N \ \text{bits/sample}$$

$$= \ l_k f_s \, \text{bits/second}$$

where $l_k$is the length of the codeword, N is the number of amplitude levels chosen and f_s is the sampling frequency.

Scalar quantization

- In scalar quantization, each sample is quantized into a level out of a finite number of levels, which is then encoded into a binary codeword.

- In fact, quantization can be interpreted as a rounding process in which each sample is rounded to the nearest value from a finite set of levels.

- The set of real numbers $\mathbb{R}$ is partitioned into N disjoint subsets denoted by $\mathcal{R}_k$, $1 \leq k \leq N$, called a quantization region.

- Corresponding to each $\mathcal{R}_k$ a quantization level $\hat{x}_k$ is chosen. If the sample at a time instant x_k belongs to $\mathcal{R}_k$ then it is represented by $\hat{x}_k$.

- Then, $\hat{x}_k$ is encoded into a binary codeword and transmitted.

- Given a number of quantization levels, we employ $\log_2 N$ bits to encode these levels in binary codewords, resulting in the rate

$$\mathrm{R} = \log_2 N \text{ bits/sample}$$

- As a result, quantization distortion is introduced and can be measured.

- The quantization procedure can be mathematically described by

$$\hat{x}_k = Q[x_k], \qquad \text{for all } x_k \in \mathcal{R}_k$$

- A distortion measure to quantify the loss of information due to quantization can be employed.

- A widely used distortion measure is the squared error distortion:

$$\begin{aligned} d(x_k, \hat{x}_k) &= (x_k - \hat{x}_k)^2 \\ &= (x_k - Q(x))^2 = e, \end{aligned}$$

where x_k is the sample to be quantized and $\hat{x}_k = Q(x_k)$ is the quantized value.

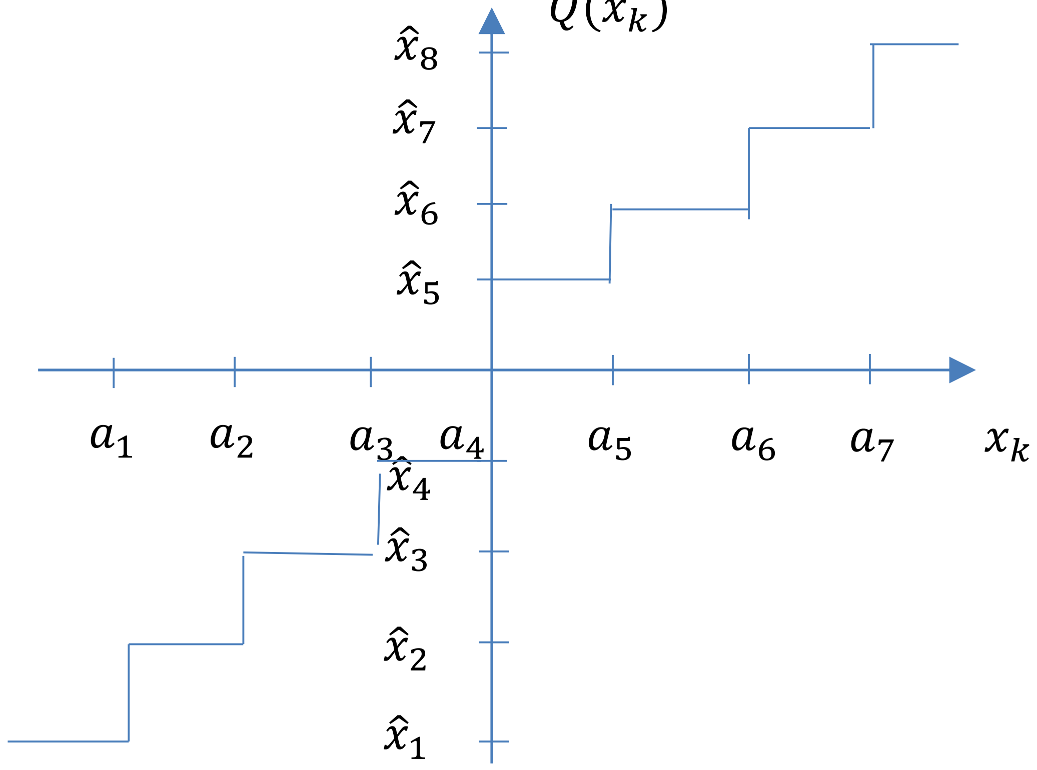
- Another distortion measure that treats x as a random variable is the mean-square error (MSE) distortion given by

$$\begin{aligned} D = E[d(x_k, \hat{x}_k)] &= E[(x_k - \hat{x}_k)^2] \\ &= E[(x_k - Q(x))^2], \end{aligned}$$

where $\hat{x}_k = Q(x_k)$ is the quantized value.

- The figure below illustrates an eight-level quantization scheme, where eight regions are defined by

$\mathcal{R}_1 = (-\infty, a_1], \mathcal{R}_2 = (a_1, a_2], \mathcal{R}_3 = (a_2, a_3], \mathcal{R}_4 = (a_3, a_4], \mathcal{R}_5 = (a_4, a_5], \mathcal{R}_6 = (a_5, a_6], \mathcal{R}_7 = (a_6, a_7]$ and $\mathcal{R}_8 = (a_7, \infty\,]$.



Example 6

Consider a sequence of samples $x_k = \{0.8; -0.3; 0.6; 0.9; 0.2; -0.15; -0.7\}$ that is quantized by a 3-bit scalar quantizer with the quantization levels contained in the following dictionary:

$$D = \{1; 0.75; 0.5; 0.25; -0.25; -0.5; -0.75; -1\}$$

Compute the quantized sequence $\hat{x}_k$ assuming that the distortion criterion is the squared error.

The quantized sequence is obtained by computing the squared error between the samples of x_k and the quantization levels $\hat{x}_k$ in the dictionary.

$$x_k = \{0.8; -0.3; 0.6; 0.9; 0.2; -0.15; -0.7\}$$

This is carried out by choosing for each sample the quantization level that yields the smallest squared error:

$$\hat{x}_k = Q(x_k) = \arg\min_D (x_k - Q(x_k))^2$$

where $D = \{1; 0.75; 0.5; 0.25; -0.25; -0.5; -0.75; -1\}$

The resulting quantized sequence is given by

$$\hat{x}_k = \{0.75; -0.25; 0.5; 1; 0.25; -0.25; -0.75\}$$

Types of scalar quantizers

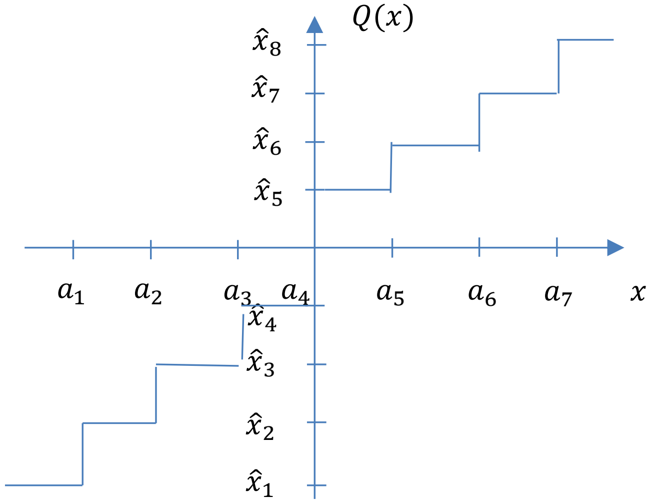
- Uniform: the quantization regions are uniform
- Non-uniform: the quantization regions are non-uniform and should match the signal's characteristics.
- Adaptive: can adapt to variations in the signal's statistics.
- Optimum: requires the pdf of the signal and an iterative numerical optimization procedure.

Uniform quantization

- Uniform quantizers are the simplest scalar quantizers where the decision regions are partitioned equally, except for the extreme regions.

- Consider a uniform quantizer with N regions of $\mathbb{R}$, where all regions except $\mathcal{R}_1$ e $\mathcal{R}_N$, have equal length equal to Δ, known as resolution.

- This means that for all $1 \leq k \leq N-2$, we have $\Delta = a_{k+1} - a_i$ and that the quantization levels are at a distance of $\frac{\Delta}{2}$ from the boundaries $a_1, a_2, \dots, a_{N-1}$.



- In a uniform quantizer, the MSE distortion is given by

$$D = \int_{-\infty}^{a_1} \left(X - \left(a_1 - \frac{\Delta}{2}\right)\right)^2 p_x(X)dX + \sum_{k=1}^{N-2} \int_{a_{1+(k-1)\Delta}}^{a_{1+k\Delta}} \left(X - \left(a_{1+k\Delta} - \frac{\Delta}{2}\right)\right)^2 p_x(X)dX$$
$$+ \int_{a_1+(N-2)\Delta}^{\infty} \left(X - \left(a_1 + (N-2)\Delta + \frac{\Delta}{2}\right)\right)^2 p_x(X)dX,$$

where D is a function of a_1 and Δ.

- In order to design an optimal uniform quantizer, we differentiate D with respect to these variables and find the values that minimize D.

- If we assume that $p_x(X)$ is an even function of x then the optimal quantizer will have symmetry properties.

- Therefore, for even N, we will have

$$a_k = -a_{N-k} = -\left(\frac{N}{2} - k\right)\Delta, \qquad \text{for } 1 \le k \le \frac{N}{2}$$

$$a_{\frac{N}{2}} = 0 \text{ and } \hat{x}_k = \hat{x}_{N+1-k}, \quad \text{for } 1 \le k \le \frac{N}{2}$$

- In this case, the distortion D is given by

$$D = \int_{-\infty}^{\left(-\frac{N}{2}-1\right)\Delta} (X - \hat{x}_1)^2 p_x(X)dX + 2\sum_{k=1}^{\frac{N}{2}-1} \int_{\left(-\frac{N}{2}+k\right)\Delta}^{\left(-\frac{N}{2}+k+1\right)\Delta} (X - \hat{x}_{k+1})^2 p_x(X)dX$$

- In these cases, minimization of distortion is often done by numerical techniques.

- The table below shows the optimal quantization level spacing for a zero-mean unit variance Gaussian random variable when $\hat{x}_k$ are chosen as mid-points of the quantization regions

Number of levels (N)	Resolution Δ	MSE (D)
1	-	1.0
2	1.596	0.3634
4	0.9957	0.1188
8	0.5860	0.03744
16	0.3352	0.01154

J. Max, "Quantizing for Minimum Distortion," IEEE Trans. Information Theory, vol. 6, no. 1, pp. 7-12, March 1960.

Example 7

Consider a signal $x(t)$ modelled as a Gaussian stochastic process with zero mean and power spectral density $S_x = \begin{cases} 2, & |f| < 100Hz \\ 0, & \text{otherwise} \end{cases}$.

The signal is sampled at the Nyquist rate and each sample is quantized using an eight level uniform quantizer with $a_1 = -60$, $a_2 = -40$, $a_3 = -20$, $a_4 = 0$, $a_5 = 20$, $a_6 = 40$, $a_7 = 60$, $\hat{x}_1 = -70$, $\hat{x}_2 = -50$, $\hat{x}_3 = -30$, $\hat{x}_4 = -10$, $\hat{x}_5 = 10$, $\hat{x}_6 = 30$, $\hat{x}_7 = 50$ and $\hat{x}_8 = 70$.

a) What is the resulting rate?

b) Compute the MSE distortion

Solution:

a) The Nyquist rate is given by

$$f_s = 2f_{\max} = 200\ Hz$$

Each sample is a zero-mean Gaussian random variable with variance

$$\sigma^2 = E\left[x_k^2\right] = R_x(0) = \int_{-\infty}^{\infty} S_x(f)df = \int_{-100}^{100} 2df = 400$$

Since each sample is quantized to 8 levels, we have that $\log_2 8 = 3$ bits are sufficient to encode each sample. Therefore, the rate is

$$R = \log_2 8\, f_s = 600 \text{ bits/s}$$

b) To find the MSE distortion, we evaluate

$$D = E[(x - \hat{x})^2] = \int_{-\infty}^{\infty} \left(X - Q(x)\right)^2 p_x(X) dX = \sum_{k=1}^{8} \int_{\mathcal{R}_k} (X - Q(x))^2 p_x(X) dX$$

$$= \int_{-\infty}^{a_1} (X - \hat{x}_1)^2 p_x(X) dX + \sum_{k=2}^{7} \int_{a_{k-1}}^{a_k} (X - \hat{x}_k)^2 p_x(X) dX + \int_{a_7}^{\infty} (X - \hat{x}_8)^2 p_x(X) dX$$

$$= 33.4$$

Signal-to-quantization noise ratio

- If the random variable x is quantized using $\mathrm{Q}(x)$ than the signal-to-quantization noise ratio (SQNR) is defined by

$$\text{SQNR} = \frac{E\left[x_k^2\right]}{E[(x_k - Q(x_k))^2]} = \frac{P_x}{P_e}$$

- The quantization noise power is given by

$$P_e = \lim_{T \to \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} E[(x_k - Q(x_k))^2]\, dt$$

- The signal power is given by

$$P_x = \lim_{T \to \infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} E\left[x_k^2(t)\right] dt$$

Example 8

Compute the SQNR for the quantization scheme of the previous example.

Solution:

Since we have $P_x = 400$ and $P_e = D = 33.4$, we obtain

$$\text{SQNR} = \frac{E\left[x_k^2\right]}{E[(x_k - Q(x_k))^2]} = \frac{P_x}{P_e} = \frac{400}{33.4}$$

In dB, we have

$$\text{SQNR}_{dB} = 10\log 10\ \text{SQNR} = 10.78\ \text{dB}$$

Vector quantization

- The idea of vector quantization is to employ blocks of samples of length n and design the quantizer in the n-dimensional Euclidean space.

- This translates into improved performance if the samples are correlated.

- Let us assume that the quantization regions in the n-dimensional Euclidean space are denoted by $\mathcal{R}_k$, $1 \leq k \leq N$.

- These N regions partition the n-dimensional space and each block of samples of length n is denoted by the n-dimensional vector $\boldsymbol{x}_k \in \mathbb{R}^n$.

Gray, R.M. (1984). "Vector Quantization". *IEEE ASSP Magazine*. **1** (2): 4–29.

- Vector quantization works as follows:

$$\text{If } \boldsymbol{x}_k \in \mathcal{R}_k$$

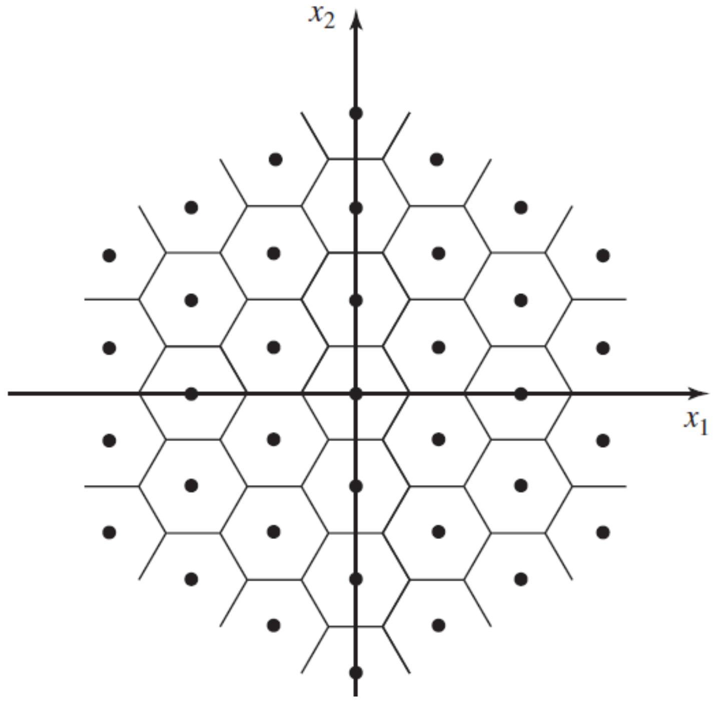
$$\text{Then } \hat{\boldsymbol{x}}_k = Q(\boldsymbol{x}_k)$$

- Since there are a total of N quantized values, $\log_2 N$ bits are enough to represent these values.

- This means that we require $\log_2 N$ bits per n source outputs, which yields the rate

$$R = \frac{\log_2 N}{n} \text{ bits / sample}$$

- An example of a vector quantizer with $n = 2$ is given below.



- The optimal vector quantizer of dimension n and N levels chooses the regions $\mathcal{R}_k$ and the quantized values $\widehat{\boldsymbol{x}}_k$ such that the resulting distortion is minimized.

- Therefore, we employ the following criteria for an optimal vector quantizer design:

i) Region $\mathcal{R}_k$ is the set of all points in the n-dimensional space that are closer to $\widehat{\boldsymbol{x}}_k$ than any other $\widehat{\boldsymbol{x}}_j$, for $\mathrm{j} \neq k$, i.e.,

$$\mathcal{R}_k = \left\{ \boldsymbol{x}_k \in \mathbb{R}: \|\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k\|^2 < \left\|\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_j\right\|^2, \forall\, \mathrm{j} \neq k \right\}$$

ii) $\widehat{\boldsymbol{x}}_l$ is the centroid of the region $\mathcal{R}_k$, i.e.,

$$\widehat{\boldsymbol{x}}_k = \frac{1}{P(\boldsymbol{x}_k \in \mathcal{R}_k)} \int \dots \int_{\mathcal{R}_k} \boldsymbol{X} p_{\boldsymbol{x}}(\boldsymbol{X}) d\boldsymbol{X}$$

- In the design of optimal vector quantizers, we start with a given set of quantization regions.

- Then, we obtain the optimal quantized vectors for these regions (criterion ii)).

- We repartition the space (criterion i)) and iterate until the changes in the distortion D are negligible.

- Algorithms such as LBG and k-means are used for this purpose and have found applications in multimedia and machine learning.

- For a vector quantizer with fixed n, the rate per vector is given by

$$B = \log_2 N \text{ bits/vector}$$

- The rate per sample is described by

$$R = \frac{B}{n} = \frac{\log_2 N}{n} \quad \text{bits/sample}$$

Example 9

Consider a sequence of 20 samples of a speech signal that is sampled at the Nyquist rate using a scalar quantizer. The number of bits per sample has to be equal to or greater than 1.

a) Compute the rate of a PCM encoder (ITU G,.711) that uses 8 bits / sample

b) Compute the rate of a vector quantizer that employs 10 bits / vector.

Solution:

a) The rate of PCM is

$$\begin{aligned} R &= 8 \text{ bits/sample} \\ &= l_k f_s = 8\ x\ 8000 = 64\ kbps \end{aligned}$$

b) The rate of the vector quantizer is

$$\begin{aligned} R = \frac{\log_2 N}{n} = \frac{10}{20} &= 0.5 \text{ bits /sample} \\ &= 0.5 \text{ x } 8000 = 4\text{kbps} \end{aligned}$$