An Introduction to Data Sciences

Gabriel Peyré
CNRS & DMA, École Normale Supérieure, PSL

CHAPTER 1

Claude Shannon and data compression

September 9, 2026

How many bits are needed to store a message without losing information? Shannon's theory relates this question to the probabilities of the symbols in the message. Starting from a small grayscale image, we introduce binary codes, explain unambiguous decoding, and derive the role of entropy in compression.

1.1 Digital data and coding

Text, sound, images and video are stored and transmitted as sequences of bits: binary digits 0 and 1. Before this binary representation is chosen, the data are described by *symbols*: letters in a text, for instance, or pixel values in an image. A lossless compression method seeks a compact binary representation of a sequence of symbols from which the original sequence can be recovered exactly.

Shannon's work [6, 7] established fundamental limits for representing and communicating information. It is useful to distinguish the operations involved:

- *Sampling* records a continuous signal at selected locations or times. *Quantization* then replaces its values by elements of a finite set.
- *Source coding*, the subject of this chapter, seeks a compact representation of the resulting symbols.
- *Channel coding* adds redundancy to detect or correct transmission errors. Protection against malicious alteration is a separate security problem.

Sampling and quantization may discard information. The coding studied here is lossless relative to the symbols that have already been acquired. This distinction matters when discussing the compression of photographs or sound recordings.

1.2 Encoding and decoding

1.2.1 A small grayscale image

A grayscale image is an array of pixels. For simplicity, consider four possible values: 0 for black, 1 for dark gray, 2 for light gray and 3 for white. The same ideas apply to a larger alphabet, such as the 256 values of an eight-bit grayscale image, and to the channels of a color image.

Figure 1.1 shows a flower and a magnified 5×5 patch. Reading its 25 values in the indicated order gives

$$(0, 1, 3, 2, 0, 3, 2, 2, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1). \tag{1.1}$$

1.2.2 Fixed-length coding

A *codeword* is the bit string assigned to a symbol. We require nonempty codewords unless the message length is known in advance. One possible assignment is

$$0 \mapsto 00, \qquad 1 \mapsto 01, \qquad 2 \mapsto 10, \qquad 3 \mapsto 11.$$

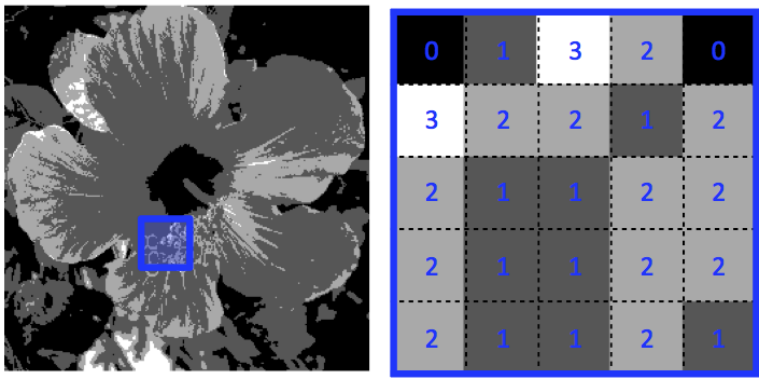


Figure 1.1. A four-level grayscale image and a magnified 5×5 patch.

All four codewords have length two, so this is a *fixed-length* code, also called a uniform code. Encoding a message means concatenating its codewords. For example,

$$(0, 1, 3) \longmapsto (00, 01, 11) \longmapsto 000111.$$

The complete message (1.1) becomes

$$\texttt{0001111000111010011010010110101001011010100101011001}.$$

Its total length is $\overline{L} = 25 \times 2 = 50$ bits. This count concerns the encoded pixel values; a file may also store dimensions, a coding table and other metadata.

For an alphabet of $N \geq 2$ symbols, a fixed-length binary code of length ℓ exists precisely when $2^\ell \geqslant N$: there must be at least as many distinct bit strings as symbols. The smallest possible length is therefore

$$\ell = \lceil \log_2 N \rceil.$$

When $N = 2^\ell$, no bit strings are left unused. For $N = 8$, for example, the codewords have length $\ell = 3$ and are

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

The base-two logarithm is defined for $x > 0$ by $\log_2 x = \ln x / \ln 2$. It satisfies $\log_2(1/x) = -\log_2 x$; in particular,

$$\log_2(1/4) = -\log_2 4 = -2.$$

The ceiling in the formula for ℓ is essential when N is not a power of two.

1.2.3 Variable-length coding

A code can use fewer bits on average by assigning shorter words to frequent symbols. Consider

$$0 \mapsto 001, \qquad 1 \mapsto 01, \qquad 2 \mapsto 1, \qquad 3 \mapsto 000. \tag{1.2}$$

Now $(0, 1, 3)$ becomes 00101000. The complete message becomes

$$\texttt{0010100010010001101110101111010111101011101},$$

which has 42 bits. The *average length per symbol* is

$$L = \frac{\overline{L}}{25} = \frac{42}{25} = 1.68 \text{ bits}.$$

This code improves on the fixed-length code for this particular message. It need not do so for every message: a sequence consisting mainly of symbols 0 and 3 would require nearly three bits per symbol.

1.2.4 Prefix codes and unambiguous decoding

A code is *uniquely decodable* if two different finite symbol sequences never produce the same bit string. Fixed-length codes with distinct codewords have this property: split the message into blocks of ℓ bits, then read the coding table backwards.

Distinct codewords alone are not sufficient for a variable-length code. For example, the assignment

$$0 \mapsto 0, \quad 1 \mapsto 10, \quad 2 \mapsto 110, \quad 3 \mapsto 101$$

is ambiguous because $1010 = 10\,10 = 101\,0$ represents either $(1, 1)$ or $(3, 0)$.

A sufficient condition for unique decoding is that no codeword be a proper prefix of another. Such a code is called a *prefix code*. The code (1.2) has this property. Its words can be recognized as soon as their last bit is read:

$$\underbrace{001}_{0}\ \underbrace{01}_{1}\ \underbrace{000}_{3}\ \underbrace{1}_{2}\ \cdots .$$

The spaces shown here explain the decoding; they are not transmitted.

The prefix condition is sufficient, but not necessary. The code $A \mapsto 0$, $B \mapsto 01$ is uniquely decodable although 0 prefixes 01. Every 1 must be paired with its preceding 0, so the complete message has only one parsing. A decoder must nevertheless look ahead after reading a 0 to decide whether it represents A or begins B. More general non-prefix codes can require delayed decisions or backtracking; prefix codes avoid this difficulty.

1.2.5 Codes as binary trees

The empty word is the root of a binary tree. Appending a 0 or a 1 selects its left or right child. A word of length ℓ lies at depth ℓ, so the complete tree of depth ℓ contains 2^ℓ leaves and $\ell + 1$ levels, including the root.

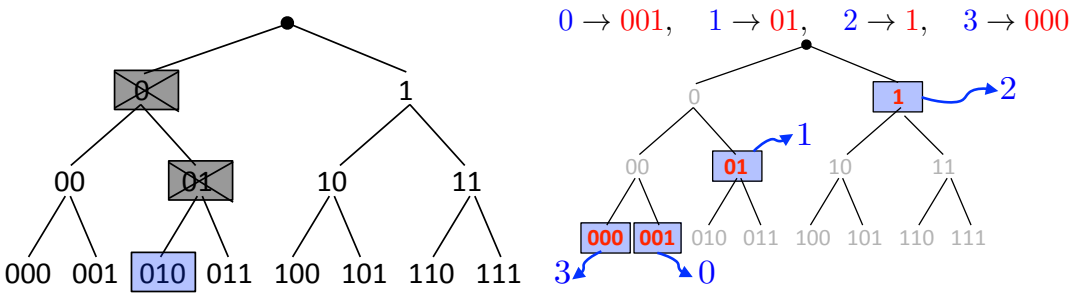


Figure 1.2. The complete binary tree to depth three (left) and the subtree of a prefix code (right).

For a prefix code, no selected codeword is an ancestor of another. The selected nodes are therefore leaves of the tree formed by their paths from the root. To decode, start at the root and follow one edge per bit. On reaching a codeword leaf, output its *symbol* and return to the root. This procedure explains why prefix decoding is immediate and unambiguous.

1.3 Entropy and the Shannon bound

1.3.1 A probabilistic model

The quality of a code depends on the messages to which it is applied. A probabilistic model makes this dependence explicit. Let V be a random symbol taking values in $\{0, \dots, N-1\}$, with probabilities

$$p_v = \mathbb{P}(V = v), \qquad p_v \geqslant 0, \qquad \sum_{v=0}^{N-1} p_v = 1.$$

The simplest model generates a sequence of independent symbols with this same distribution. Independence is an assumption about the source, not a property guaranteed by the coding algorithm.

For an observed message, probabilities can be estimated by *empirical frequencies*. The message (1.1) contains 2 zeros, 9 ones, 12 twos and 2 threes. Its empirical distribution is

$$p = (2/25,\ 9/25,\ 12/25,\ 2/25).$$

Using these frequencies as probabilities is a modeling choice. If the decoder does not already know the chosen model or coding table, the information needed to describe it must also be transmitted.

1.3.2 Entropy

The *entropy* of V, measured in bits, is

$$H(V) = -\sum_{v=0}^{N-1} p_v \log_2 p_v. \tag{1.3}$$

A symbol with probability $p_v > 0$ carries the information $-\log_2 p_v$. Entropy is the average of this quantity. The convention $0 \log_2 0 = 0$ means that impossible symbols contribute nothing; it agrees with $\lim_{x \downarrow 0} x \log_2 x = 0$.

Since $-\log_2 p_v \geqslant 0$, entropy is nonnegative. More precisely,

$$0 \leqslant H(V) \leqslant \log_2 N.$$

The endpoints describe two useful models:

- **No uncertainty.** If $p_1 = 1$, every symbol equals 1 and $H(V) = 0$. The message is a sequence of ones.
- **Maximum uncertainty.** If $p_v = 1/N$ for all v, then $H(V) = \log_2 N$. With four equally likely symbols, this is two bits. A random message from this model need not contain exactly the same number of each symbol.
- **An intermediate case.** For the empirical distribution above,

$$H(V) = -2\frac{2}{25}\log_2\frac{2}{25} - \frac{9}{25}\log_2\frac{9}{25} - \frac{12}{25}\log_2\frac{12}{25} \simeq 1.6219 \text{ bits}.$$

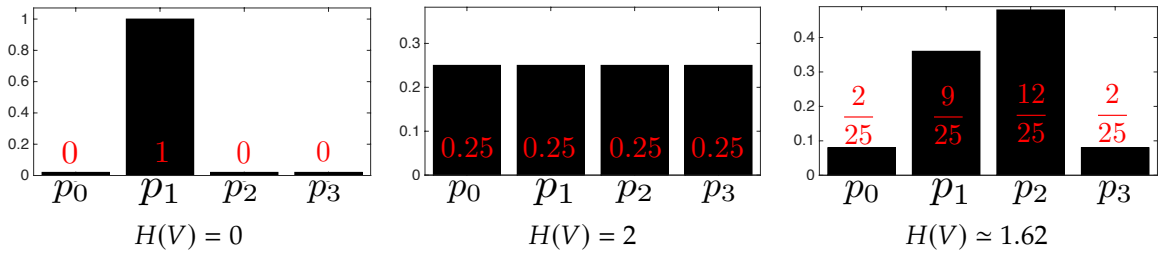


Figure 1.3. A deterministic distribution, the uniform distribution on four symbols, and the empirical distribution of the 25-pixel message.

1.3.3 Expected code length

Let c_v be the codeword for v and ℓ_v its length. The expected number of bits per symbol is

$$L(c) = \mathbb{E}[\ell_V] = \sum_v p_v \ell_v.$$

For the code (1.2), the lengths are $(3, 2, 1, 3)$, hence

$$L(c) = \frac{2 \times 3 + 9 \times 2 + 12 \times 1 + 2 \times 3}{25} = \frac{42}{25} = 1.68.$$

If p is the empirical distribution of a finite message, this formula is also its exact average code length. If p is a source model, it is an expectation.

1.3.4 The entropy lower bound

Every binary prefix code satisfies the Shannon bound [6]

$$H(V) \leqslant L(c). \tag{1.4}$$

Here is a short proof. Let m be the greatest codeword length. A word of length ℓ_v has $2^{m-\ell_v}$ descendants at depth m. Prefix-free words have disjoint sets of descendants, so

$$\sum_v 2^{m-\ell_v} \leqslant 2^m, \qquad Z \stackrel{\text{def.}}{=} \sum_v 2^{-\ell_v} \leqslant 1.$$

This is Kraft's inequality. Define $q_v = 2^{-\ell_v}/Z$. Then $\sum_v q_v = 1$ and $\ell_v = -\log_2 q_v - \log_2 Z$, which gives

$$\begin{aligned} L(c) - H(V) &= \sum_{v:p_v>0} p_v \log_2 \frac{p_v}{q_v} - \log_2 Z \\ &\geqslant \sum_{v:p_v>0} p_v \log_2 \frac{p_v}{q_v} \geqslant 0. \end{aligned}$$

The last sum is the Kullback–Leibler divergence. Its nonnegativity follows from $\ln t \leqslant t - 1$: applying this to $t = q_v/p_v$ and summing yields $\sum p_v \ln(q_v/p_v) \leqslant \sum_{p_v>0} q_v - 1 \leqslant 0$.

The bound concerns an *average*; it does not impose $\ell_v \geqslant -\log_2 p_v$ separately for each symbol. The real numbers $-\log_2 p_v$ nevertheless suggest useful target lengths: a *larger* probability gives a smaller target length. For a uniform distribution on 2^ℓ symbols, the fixed-length code attains the bound exactly. For other alphabet sizes, integer codeword lengths can prevent equality.

1.3.5 Huffman coding and blocks

Huffman's algorithm [2] constructs a prefix code with minimum expected length for a given finite distribution. Symbols of probability zero are omitted from the effective alphabet. For a source with at least two positive-probability symbols, its length satisfies the strict upper bound

$$H(V) \leqslant L(c) < H(V) + 1.$$

One way to understand the upper bound is to round each target length up: $\ell_v = \lceil -\log_2 p_v \rceil$. These lengths satisfy Kraft's inequality and can be realized by a prefix code. Their average is less than $H(V) + 1$, and a Huffman code does at least as well.

To reduce the overhead per symbol, encode blocks of n independent symbols. Their entropy is $H(V_1, \ldots, V_n) = nH(V)$, so a suitable block code has expected length L_n satisfying

$$H(V) \leqslant \frac{L_n}{n} < H(V) + \frac{1}{n}.$$

Thus the rate approaches entropy as the block length grows. Arithmetic coding [5] implements related ideas without storing an exponentially large table of block codewords. Finite messages still incur model, termination and numerical-precision overheads.

If a source has only one possible symbol, its entropy is zero. When this symbol and the message length are already known, no payload bits are needed. Requiring a nonempty codeword would instead spend at least one bit per symbol. The convention about framing and the effective alphabet must therefore be stated.

1.3.6 Exploiting dependence between pixels

Neighboring pixels in an image are usually dependent. A symbol-by-symbol model ignores this structure. A useful remedy is to transform the data before coding them.

Index the P pixels by $v_0, \ldots, v_{P-1}$ and define

$$d_i = v_i - v_{i-1}, \qquad 1 \leqslant i \leqslant P - 1.$$

Keep v_0 as well as the differences. The transformation

$$(v_0, \ldots, v_{P-1}) \longmapsto (v_0, d_1, \ldots, d_{P-1})$$

is bijective, since $v_i = v_0 + \sum_{j=1}^{i} d_j$. In a nearly constant region, most differences are zero or small. For four-level pixels, they lie in $\{-3, -2, -1, 0, 1, 2, 3\}$.

For the flower example, the difference histogram is more concentrated than the pixel histogram. The empirical entropies and Huffman lengths shown in Figure 1.5 illustrate the resulting gain when values are coded separately. This gain is not guaranteed for every image.

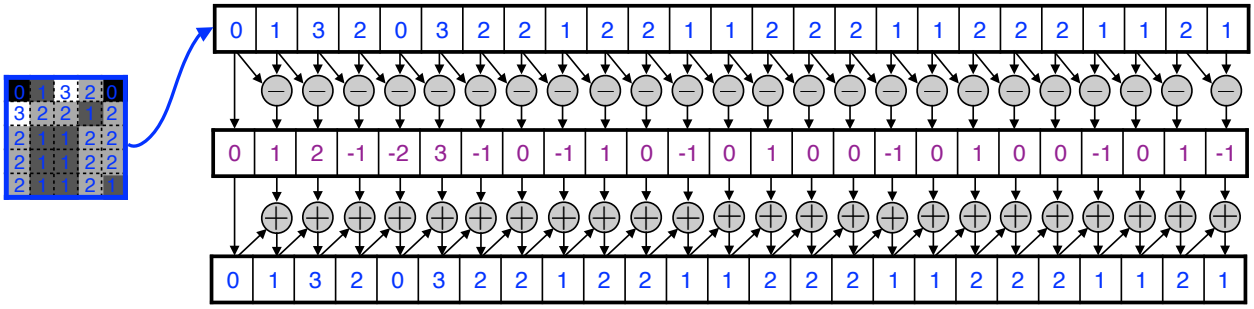
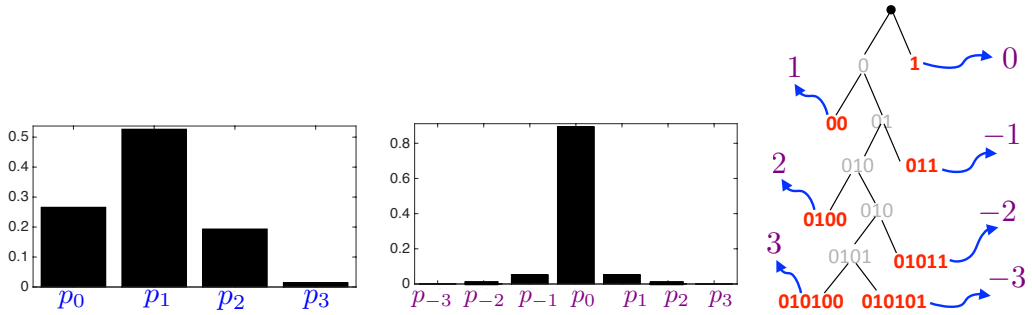


Figure 1.4. Successive differences and reconstruction by cumulative summation, retaining the first pixel.



$H \simeq 1.54,\ L \simeq 1.67$ $H \simeq 0.61,\ L \simeq 1.16$ Prefix tree

Figure 1.5. Pixel and difference histograms over the entire image, with a prefix-code tree for the differences.

The displayed difference code is

d	-3	-2	-1	0	1	2	3
c_d	010101	01011	011	1	00	0100	010100

and has an average length of approximately 1.16 bits per difference. Using the displayed rounded rates for a 256×256 image gives payload estimates of 16 KiB for the two-bit fixed-length code, 13.36 KiB for direct pixel coding, and 9.28 KiB for difference coding. Here 1 KiB is 1024 bytes, or 8192 bits. The initial pixel and coding-table overhead must be added to the last estimate.

A bijective transformation does *not* reduce the joint entropy of the whole message. It can make separate-symbol models more effective by changing the marginal distributions and the dependence structure. The bound (1.4) itself needs no independence assumption for a single random symbol; independence was used above to obtain the block identity $H(V_1, \dots, V_n) = nH(V)$. For a dependent source, the joint entropy of the block is the relevant lower bound.

More advanced image codecs combine prediction or transforms with quantization and entropy coding. JPEG 2000, for example, uses wavelets and supports both lossless and lossy modes [3]. The same principle of exploiting dependence also applies to the letters of a text.

Further reading. For further details on source coding, see [1], and [3] for signal and image processing. The original figure programs are available at https://github.com/gpeyre/2016-shannon-theory; related experiments appear in the Numerical Tours [4].

Glossary

- **Pixel**: a location in an image grid, or its stored value.
- **Symbol**: an element of the finite alphabet to be encoded.
- **Codeword**: the bit string assigned to one symbol.
- **Code**: the assignment of codewords to symbols.
- **Empirical distribution**: the observed relative frequencies of symbols.
- **Histogram**: a graphical representation of these frequencies.
- **Source model**: a probability law for the symbols or their sequences.
- **Entropy**: the nonnegative average information $H(V)$, in bits.
- **Average code length**: $L(c) = \sum_v p_v \ell_v$, in bits per symbol.

Acknowledgements. I thank Marie-Noëlle Peyré, Gwenn Guichaoua, François Béguin, Gérard Grancher, Aurélien Djament and François Sauvageot for their careful reading of the French text. The flower photograph is by Maïtine Bergounioux.

References

[1] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, 2006.

[2] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, 1952.

[3] S. G. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier/Academic Press, Amsterdam, third edition, 2009.

[4] Gabriel Peyré. The numerical tours of signal processing. *Computing in Science & Engineering*, 13(4):94–97, 2011. doi: 10.1109/MCSE.2011.71. URL https://www.numerical-tours.com/.

[5] J. Rissanen and G. Langdon. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162, 1979.

[6] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[7] C. E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, 1949.