Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA, École Normale Supérieure

CHAPTER 3

Shannon Coding Theory

September 9, 2026

A sequence of symbols can often be stored using fewer bits when its probabilities and dependencies are taken into account. The challenge is to reduce the expected length while keeping every message unambiguously recoverable. Prefix codes and entropy lead to Shannon's source coding bound; Huffman's construction and block coding show how to approach it. Empirical block counts connect these bounds to finite texts, while models of dependent symbols reveal further compression gains.

3.1 Source Coding

Uniform coding. We consider an alphabet $(s_1, \dots, s_K)$ of K symbols. For signal samples $0 \leqslant f_0 < 1$ quantized into K bins, the symbols label the bins. For text, they may represent letters, punctuation marks, and spaces. A fixed-length binary code uses $\lceil \log_2(K) \rceil$ bits per symbol. For example, if $K = 4$ and the symbols are $\{0, 1, 2, 3\}$, one possible code is $(c_0 = 00, c_1 = 01, c_2 = 10, c_3 = 11)$.

When some symbols are much more frequent than others, variable-length codes can reduce the average number of bits per symbol by assigning shorter codewords to frequent symbols.

Prefix coding. A binary code assigns each symbol s_k a finite word $c_k = c(s_k) \in \bigcup_{\ell \geqslant 0} \{0,1\}^\ell$ of length $|c_k|$. An empty word is allowed only for a one-symbol alphabet, with the message length supplied separately. A code is a prefix code if no codeword is a prefix of another. Equivalently, the codewords correspond to leaves of a binary tree T: starting at the root, traversing an edge to the left appends 0, and traversing an edge to the right appends 1. We write $c = \text{Leaves}(T)$ for this tree representation.

To decode a binary stream, follow its bits down the tree. Upon reaching a leaf, output the corresponding symbol and return to the root. The prefix property makes this decoding unambiguous and instantaneous: each symbol is recognized as soon as the last bit of its codeword has been read.

For codes with nonempty codewords, *unique decodability* means that every encoded finite message has exactly one decomposition into codewords. Prefix codes have this property, but a uniquely decodable code need not be a prefix code. For example, consider

$$c(s_1) = 0, \qquad c(s_2) = 01.$$

The word 0 is a prefix of 01, yet decoding is unique: every 1 must be paired with the immediately preceding 0 to form 01, and every remaining 0 represents s_1. Thus 00101 can only be split as $0 \mid 01 \mid 01$, encoding $s_1 s_2 s_2$.

After reading a 0, however, the decoder must inspect the next bit, or know that the message has ended, before deciding which symbol to output. If it has already committed to s_1 and the next bit is 1, it must backtrack and decode 01 as s_2. One bit of lookahead avoids this backtracking here, but still introduces a decoding delay. Prefix coding avoids such tentative decisions and the associated lookahead or backtracking overhead.

Probabilistic modeling. We seek a code with the smallest expected length. Let $p = (p_1, \dots, p_K) \in \mathbb{R}_+^K$ be the symbol probability vector, with $\sum_k p_k = 1$. In practice, one often estimates p_k from the relative frequency of s_k

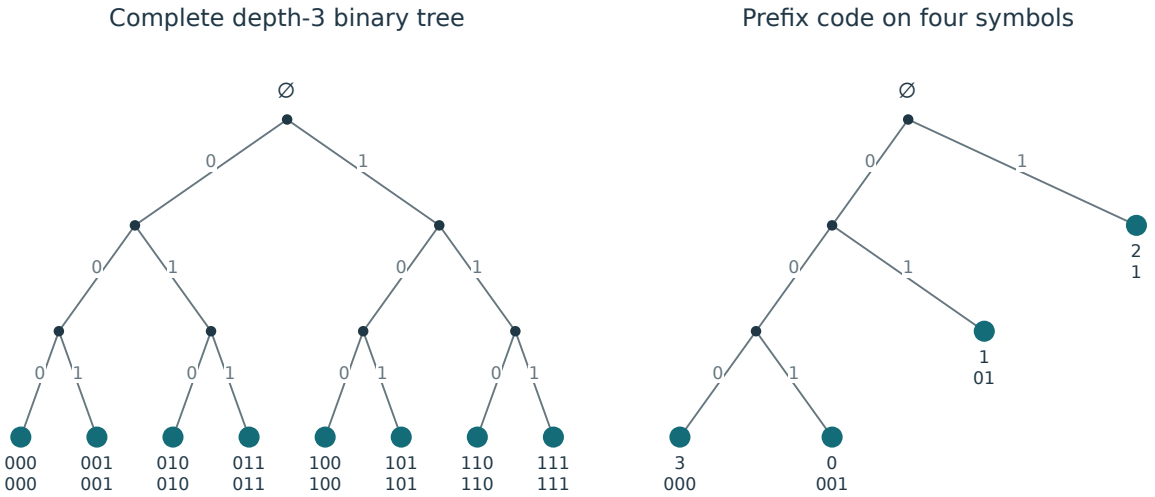

Figure 3.1. Left: the complete tree of binary words of length 3. Right: a prefix code.

in the data. The decoder must also know the resulting code, either from a shared model or from a transmitted description.

The entropy of the probability distribution p is

$$H(p) \stackrel{\text{def.}}{=} -\sum_k p_k \log_2(p_k)$$

with the convention $0 \log_2(0) = 0$.

For $h(u) = -u \log_2 u$ and $u > 0$, one has $h'(u) = -(\ln u + 1)/\ln 2$ and $h''(u) = -1/(u \ln 2) < 0$. Thus H is strictly concave on the probability simplex.

For a probability density f with respect to a reference measure $\mathrm{d}x$, such as Lebesgue measure on $\mathbb{R}^d$, differential entropy is defined by $H(f) \stackrel{\text{def.}}{=} -\int f(x) \log_2(f(x)) \, \mathrm{d}x$, whenever this integral is defined. Unlike discrete entropy, it can be negative and depends on the reference measure.

Lemma 3.1. *For every probability vector p,*

$$0 = H(\delta_i) \leqslant H(p) \leqslant H(1/K) = \log_2(K)$$

where δ_i is the point mass at symbol i, and $1/K$ denotes the uniform probability vector.

Proof. For $u \in [0, 1]$, $-u \log_2 u \geqslant 0$, so $H(p) \geqslant 0$, with equality precisely for a point mass. For probability vectors p, q with $q_i > 0$ whenever $p_i > 0$, the inequality $\ln u \leqslant u - 1$ gives

$$\sum_{i : p_i > 0} p_i \log_2 \frac{q_i}{p_i} \leqslant \frac{1}{\ln 2} \left(\sum_{i : p_i > 0} q_i - 1 \right) \leqslant 0.$$

Consequently $H(p) \leqslant -\sum_i p_i \log_2 q_i$. Taking $q_i = 1/K$ yields $H(p) \leqslant \log_2 K$, with equality precisely when p is uniform. □

The quantity $\mathrm{KL}(p \mid q) \stackrel{\text{def.}}{=} \sum_i p_i \log_2(p_i/q_i) \geqslant 0$ is called the Kullback–Leibler divergence, or relative entropy between p and q. The conventions are $0 \log_2(0/q_i) = 0$ and $p_i \log_2(p_i/0) = +\infty$ for $p_i > 0$. Although it vanishes exactly when $p = q$, it is not a metric: it is generally asymmetric and does not satisfy the triangle inequality. Minimizing the cross-entropy between an empirical distribution and a model is equivalent to maximizing the likelihood, and to minimizing their relative entropy. Cross-entropy is therefore a common objective in classification.

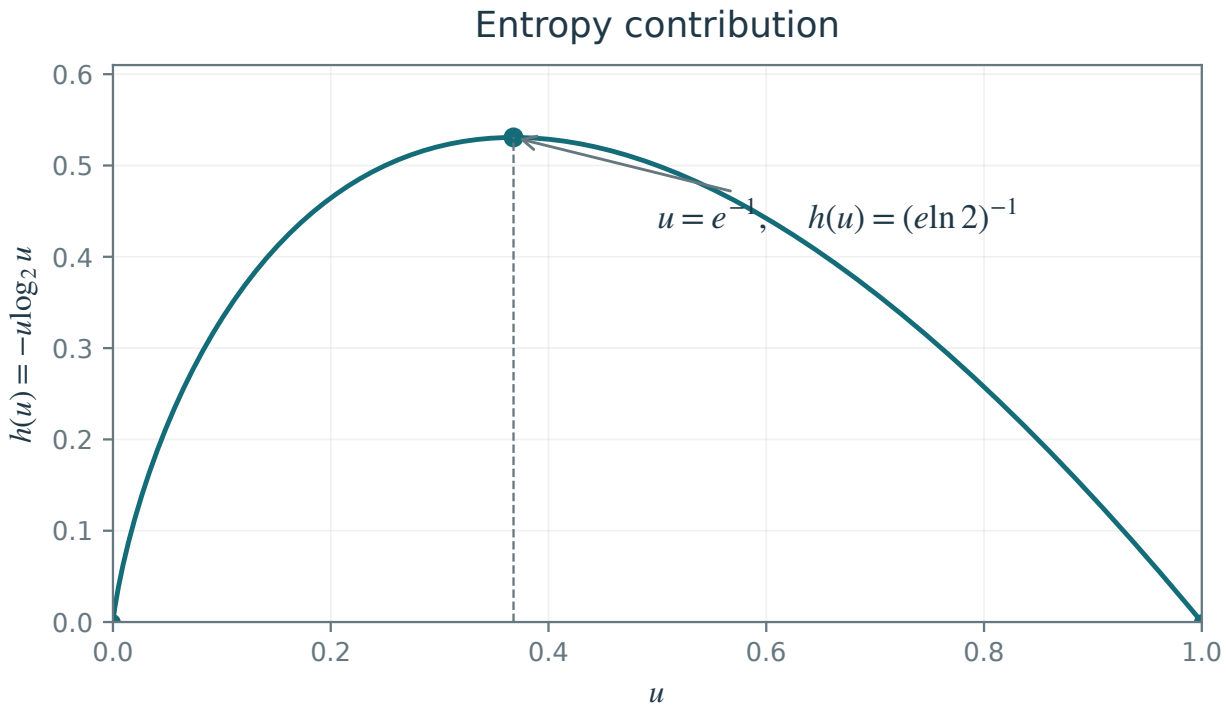

Figure 3.2. The concave function $h(u) = -u \log_2 u$ underlying entropy.

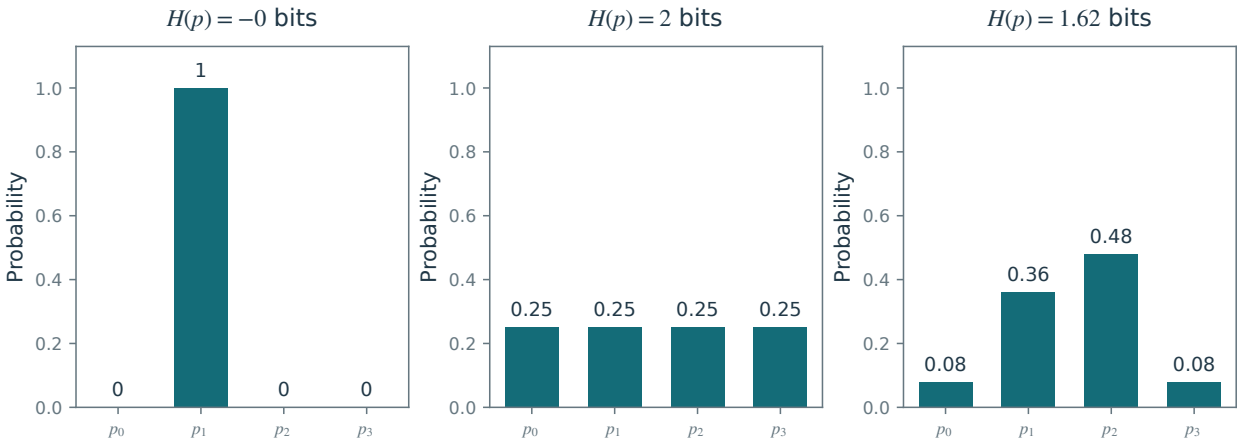

Figure 3.3. Three examples of probability distributions with corresponding entropies in bits.

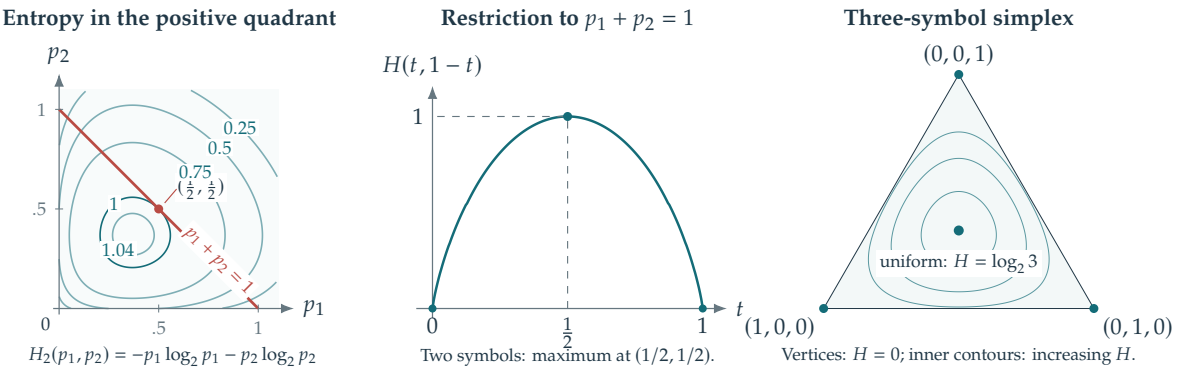

Figure 3.4. Entropy is minimized at a point mass and maximized at the uniform distribution. Left: level sets of the entropy formula in the positive quadrant and the constraint $p_1 + p_2 = 1$. Middle: its restriction to that constraint. Right: entropy level sets on the three-symbol probability simplex.

3.2 Shannon Source Coding Theorem

For a symbol distribution p, the expected codeword length of a code c is

$$L(c) \stackrel{\text{def.}}{=} \sum_k p_k |c_k|.$$

We seek a prefix code minimizing $L(c)$. Shannon's source coding theorem bounds the optimal expected length in terms of entropy.

Theorem 3.2. *Assume $p_k > 0$ for every symbol in the alphabet. Symbols of zero probability may be omitted. (i) If $c = \mathit{Leaves}(T)$ for some tree T, then*

$$L(c) \geqslant H(p).$$

(ii) Conversely, there exists a code c with $c = \mathit{Leaves}(T)$ such that

$$L(c) < H(p) + 1.$$

Kraft's inequality characterizes the possible codeword lengths of a prefix code and is the key to the proof.

Lemma 3.3 (Kraft inequality)**.** *(i) For a code c, if there exists a tree T such that $c = \mathit{Leaves}(T)$ then*

$$\sum_k 2^{-|c_k|} \leqslant 1. \tag{3.1}$$

(ii) Conversely, if $(\ell_k)_k$ are nonnegative integers such that

$$\sum_k 2^{-\ell_k} \leqslant 1 \tag{3.2}$$

then there exists a code $c = \mathit{Leaves}(T)$ such that $|c_k| = \ell_k$.

Proof. $\Rightarrow$ Suppose $c = \mathrm{Leaves}(T)$ and put $m = \max_k |c_k|$. Complete T to a full binary tree of depth m. The subtree rooted at c_k has height $m - |c_k|$ and $2^{m-|c_k|}$ leaves; see Figure 3.5, panel 1. These subtrees are disjoint, and the full tree has 2^m leaves, so

$$\sum_k 2^{m-|c_k|} \leqslant 2^m,$$

which gives (3.1) after division by 2^m.

$\Leftarrow$ Conversely, assume (3.2) holds and order the lengths so that $\ell_1 \leqslant \ldots \leqslant \ell_K =: m$. We construct full subtrees of height $m - \ell_k$, each with $2^{m-\ell_k}$ leaves, and place their leaf blocks consecutively from left to right. The inequality $\sum_k 2^{m-\ell_k} \leqslant 2^m$ ensures that all blocks fit among the 2^m leaves. Because the lengths ℓ_k are nondecreasing, the block sizes are nonincreasing powers of two. Each block therefore begins at a multiple of its own size and is the leaf set of a subtree; see Figure 3.5, panel 2. The roots of these disjoint subtrees form the required prefix code. □

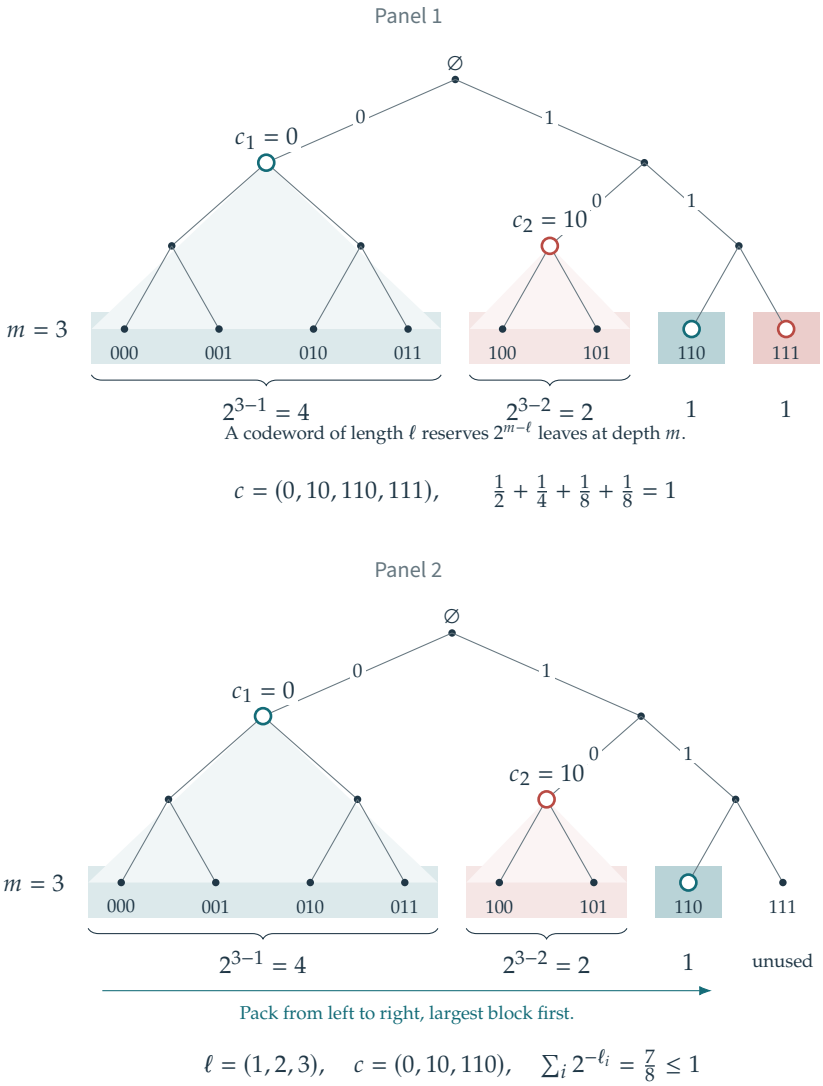

Figure 3.5. Panel 1: full binary tree obtained by completing the tree associated with the code ($c_1 = 0, c_2 = 10, c_3 = 110, c_4 = 111$). Panel 2: subtrees with the prescribed codeword lengths packed from left to right.

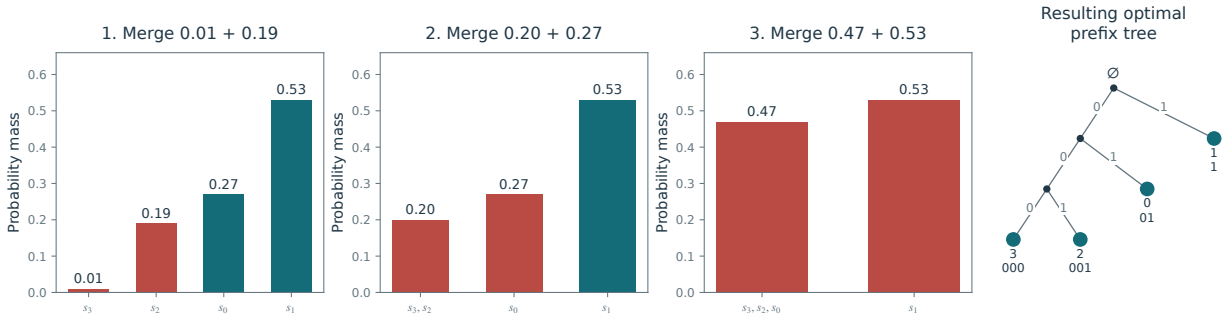

Figure 3.6. Successive merges in Huffman's coding algorithm.

Proof of Shannon's theorem. (i) Put $\ell_k = |c_k|$ and $Z = \sum_k 2^{-\ell_k}$. Kraft's inequality gives $0 < Z \leqslant 1$, so $q_k = 2^{-\ell_k}/Z$ defines a strictly positive probability vector. Since $2^{-\ell_k} = Z q_k$, we have

$$\ell_k = -\log_2(Z q_k) = -\log_2 q_k - \log_2 Z.$$

Substituting this expression and using $\sum_k p_k = 1$ gives

$$\begin{aligned} L(c) - H(p) &= \sum_k p_k \left(\ell_k + \log_2 p_k\right) \\ &= \sum_k p_k \log_2 \frac{p_k}{q_k} - (\log_2 Z) \sum_k p_k \\ &= \mathrm{KL}(p \mid q) - \log_2 Z \geqslant \mathrm{KL}(p \mid q) \geqslant 0. \end{aligned} \tag{3.3}$$

The first inequality follows from $Z \leqslant 1$, and the second from the nonnegativity of relative entropy. Equality $L(c) = H(p)$ holds exactly when $Z = 1$ and $q = p$; then $\ell_k = -\log_2 p_k$, so these ideal lengths must be integers.

(ii) Choose $\ell_k = \lceil -\log_2 p_k \rceil$. Then $\sum_k 2^{-\ell_k} \leqslant \sum_k p_k = 1$, so Lemma 3.3 constructs a prefix code with these lengths. Moreover,

$$L(c) = \sum_k p_k \lceil -\log_2 p_k \rceil < \sum_k p_k (-\log_2 p_k + 1) = H(p) + 1.$$

For a one-symbol alphabet, the empty word has length zero and the same conclusions hold. □

This proof is constructive: rounding the ideal lengths and using Kraft's lemma yields a Shannon code. It need not minimize $L(c)$. Huffman's greedy algorithm constructs an optimal binary prefix code by repeatedly merging the two least probable symbols. With a priority queue, its cost is $O(K \log K)$. Figure 3.6 illustrates the construction.

`Associated code: coding/test_text.m`

3.3 Probabilistic Modeling

3.3.1 Block Coding

Symbol-by-symbol prefix coding may be inefficient when one symbol has probability close to one: its ideal length $-\log_2 p_k$ is then below one bit, whereas a nontrivial prefix code uses at least one bit per symbol. A remedy is to group r consecutive symbols into blocks and code the resulting alphabet of size K^r.

For independent symbols with common distribution p, the block distribution is

$$P_{k_1,\dots,k_r} = p_{k_1} \cdots p_{k_r}, \qquad H(P) = rH(p).$$

Shannon's theorem applied to blocks gives an expected length per original symbol in the interval $[H(p), H(p) + 1/r)$. This is an exact expectation bound under the model; no large-sample limit is needed. The code is prefix-free on blocks, so it need not use an integer number of bits per original symbol on average.

For dependent symbols whose marginal distributions are all p, entropy subadditivity gives

$$H(P) \leqslant rH(p),$$

with equality if and only if the symbols within the block are independent. For N divisible by r, the expected total length is at most

$$\frac{N}{r}\big(H(P)+1\big) \leqslant N\left(H(p)+\frac{1}{r}\right).$$

Modeling dependence can therefore reduce the expected length further. For a stationary source on a finite alphabet, the block entropies are subadditive, so the entropy rate $\lim_{r\to\infty} H(P)/r$ exists. This rate is the asymptotic lower bound on the expected number of bits per symbol.

3.3.2 Empirical Block Coding

The same entropy bound has a finite-text counterpart. Fix a text $x_1 \cdots x_N$ over the alphabet $\mathcal{A} = \{s_1, \ldots, s_K\}$, and assume $N = mr$ with $m, r \geqslant 1$. Split it into m non-overlapping blocks of length r:

$$B_s = (x_{(s-1)r+1}, \ldots, x_{sr}) \in \mathcal{A}^r, \qquad s = 1, \ldots, m.$$

Define the empirical symbol and block distributions by counting occurrences:

$$\hat{p}(a) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\{x_i = a\}}, \qquad a \in \mathcal{A}, \tag{3.4}$$

$$\hat{P}(w) = \frac{1}{m}\sum_{s=1}^{m} \mathbf{1}_{\{B_s = w\}}, \qquad w \in \mathcal{A}^r. \tag{3.5}$$

These distributions describe the fixed text; no assumption on how it was generated is needed.

For each position $t \in \{1, \ldots, r\}$ within a block, let

$$\hat{q}_t(a) = \frac{1}{m}\sum_{s=1}^{m} \mathbf{1}_{\{x_{(s-1)r+t} = a\}} = \sum_{w : w_t = a} \hat{P}(w).$$

Thus $\hat{q}_t$ is the t-th marginal of $\hat{P}$. The position marginals need not coincide, but every symbol occurrence contributes to exactly one of them. Consequently,

$$\hat{p} = \frac{1}{r}\sum_{t=1}^{r} \hat{q}_t. \tag{3.6}$$

Proposition 3.4 (Empirical block entropy)**.** *For every fixed text and every block length r dividing N,*

$$H(\hat{P}) \leqslant \sum_{t=1}^{r} H(\hat{q}_t) \leqslant rH(\hat{p}). \tag{3.7}$$

Proof. Consider the product of the position marginals,

$$Q = \hat{q}_1 \otimes \cdots \otimes \hat{q}_r, \qquad Q(w) = \prod_{t=1}^{r} \hat{q}_t(w_t).$$

If $\hat{P}(w) > 0$, then $\hat{q}_t(w_t) > 0$ for every t, so $Q(w) > 0$. Relative entropy is therefore finite. Expanding $\log_2 Q(w)$ and summing over each coordinate gives

$$\begin{aligned} 0 \leqslant \mathrm{KL}(\hat{P} \mid Q) &= \sum_{w : \hat{P}(w) > 0} \hat{P}(w) \log_2 \frac{\hat{P}(w)}{Q(w)} \\ &= -H(\hat{P}) + \sum_{t=1}^{r} H(\hat{q}_t). \end{aligned}$$

This proves the first inequality. For the second, concavity of entropy and the counting identity (3.6) yield

$$\frac{1}{r}\sum_{t=1}^{r} H(\hat{q}_t) \leqslant H\left(\frac{1}{r}\sum_{t=1}^{r} \hat{q}_t\right) = H(\hat{p}).$$

□

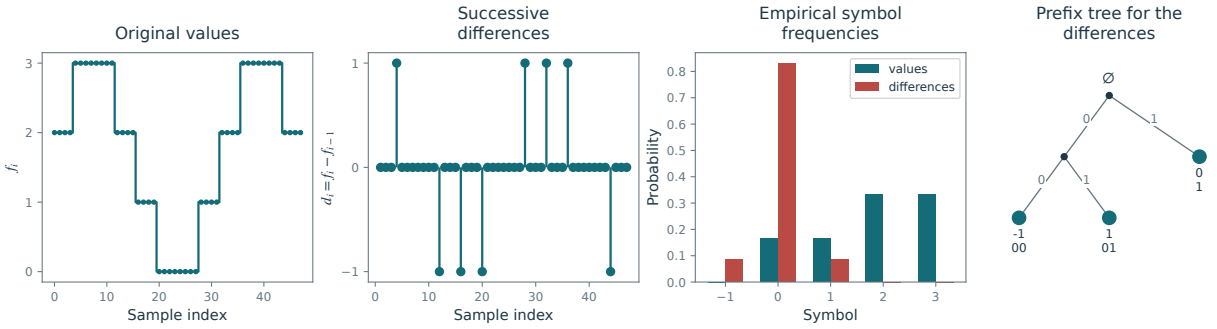

Figure 3.7. Difference coding. From left to right: the signal, its first differences, the corresponding histograms, and a code tree for the differences.

The first inequality is an equality exactly when $\hat{P} = Q$; the second is an equality exactly when all position marginals equal $\hat{p}$. Hence $H(\hat{P}) = rH(\hat{p})$ holds exactly when $\hat{P} = \hat{p}^{\otimes r}$.

Example 3.5 (Position frequencies and dependence). For the binary text 01010101 and $r = 2$, the symbol distribution is $\hat{p} = (1/2, 1/2)$, while $\hat{q}_1 = (1, 0)$ and $\hat{q}_2 = (0, 1)$. Every block is 01, so $H(\hat{P}) = 0$ although $rH(\hat{p}) = 2$. This shows why one must average the position marginals rather than identify each of them with $\hat{p}$.

For the text 0011 and $r = 2$, both position marginals equal $(1/2, 1/2)$, but only the blocks 00 and 11 occur. The dependence within each block gives $H(\hat{P}) = 1 < 2 = H(\hat{q}_1) + H(\hat{q}_2)$.

Encoded length of the fixed text. For a prefix code c on the observed blocks, the actual number of encoded bits is

$$\ell_{\text{text}}(c) = \sum_{s=1}^{m} |c(B_s)| = m \sum_{w:\hat{P}(w)>0} \hat{P}(w)|c(w)|.$$

Applying Theorem 3.2 to $\hat{P}$ shows that every such code satisfies $\ell_{\text{text}}(c) \geqslant mH(\hat{P})$, and that some code satisfies

$$\frac{\ell_{\text{text}}(c)}{N} < \frac{H(\hat{P})}{r} + \frac{1}{r} \leqslant H(\hat{p}) + \frac{1}{r}. \tag{3.8}$$

This bounds the actual average number of bits per symbol for the given text, at every admissible block length. The decoder is assumed to know the block code, r, and N; any transmitted codebook or message-length description adds to this length. In particular, when only one distinct block occurs, its empty codeword can be decoded because the number of blocks is known.

3.3.3 Mismatched Probability Models

For ideal real-valued lengths based on a model q, the excess expected length is the relative entropy, also called the Kullback–Leibler divergence. For $q_k > 0$ whenever $p_k > 0$, this excess is

$$\mathrm{KL}(p|q) = -\sum_k p_k \log_2 q_k - H(p) = \sum_k p_k \log_2 \frac{p_k}{q_k} \geqslant 0.$$

After lengths are rounded to integers, the difference between the expected lengths of two prefix codes need not equal this divergence. Relative entropy is jointly convex in (p, q) and is a central tool for comparing probability distributions in information theory and statistics.

3.4 Exploiting Statistical Dependence

A memoryless code based on the marginal distribution cannot have expected length below the marginal entropy. Dependent data, however, can be coded more efficiently by modeling their joint distribution. An invertible transformation of the discrete symbols can make the dependence easier to encode while preserving their joint entropy. For example, consecutive samples of a smooth signal often have small differences. Figure 3.7 illustrates coding these differences instead of the original values. The initial sample must also be retained to make the transformation invertible.

Another way to exploit temporal redundancy is run-length coding: each maximal run of identical symbols is represented by the symbol and the run length, which can itself be entropy-coded. Its performance depends on how the run lengths and transitions are modeled.

For a stationary finite-state Markov chain with stationary probabilities π_i and transition probabilities P_{ij}, the entropy rate is $-\sum_{i,j} \pi_i P_{ij} \log_2 P_{ij}$. Coding with the conditional transition probabilities, or coding sufficiently long blocks, can approach this rate. Run-length coding alone does not guarantee this performance.