

INFORMATION THEORY AND APPLICATIONS IN COMPUTER SCIENCE

Lecture 2: Coding, Divergence, and Counting

Draft

23 September 2026

Conventions. Alphabets are finite and logarithms are base two. Write $p(x) = \Pr[X = x]$, $\text{supp } p = \{x : p(x) > 0\}$, and $h_2(t) = -t \log t - (1-t) \log(1-t)$, with $0 \log 0 = 0$. We use $H(p)$ and $H(X)$ interchangeably when $X \sim p$. Write $X^n = (X_1, \dots, X_n)$ and $X_{<i} = (X_1, \dots, X_{i-1})$. Expectations omit zero-probability outcomes, and conditional distributions are used only at events of positive probability. Our bounds concern description length and information; they impose no running-time restriction.

2 From uncertainty to descriptions and counting

Lecture 1 gave entropy and mutual information precise algebraic meanings. We now give those quantities operational meaning. A lossless code must distinguish every possible source outcome, but it can save bits on average by assigning shorter descriptions to common outcomes. Kraft's inequality characterizes which lengths are compatible with decoding; the source coding theorem then shows that entropy is the best achievable average rate.

This coding viewpoint also explains relative entropy: using a probability model different from the true source incurs an extra expected log loss. Mutual information is a special case, measuring the cost of ignoring dependence. Finally, we reverse the viewpoint: to bound the size of a set, choose a uniform random element and bound its entropy.

The facts from Lecture 1 that we will use are

$$H(X, Y) = H(Y) + H(X | Y), \quad H(X | Y, Z) \leq H(X | Y), \quad I(X; Y) = H(X) - H(X | Y),$$

together with $H(X) \leq \log |\text{supp } p|$ and Jensen's inequality. In particular, conditioning reduces entropy *on average*; no pointwise claim about every observation is being assumed.

2.1 Variable-length descriptions need a decoding rule

Suppose a source produces one of four symbols with probabilities

$$p(a) = \frac{1}{2}, \quad p(b) = \frac{1}{4}, \quad p(c) = p(d) = \frac{1}{8}.$$

Fixed-length names use two bits each. The assignments

$$a \mapsto 0, \quad b \mapsto 10, \quad c \mapsto 110, \quad d \mapsto 111$$

instead have average length $\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 7/4$ bits. For example, the concatenation 0101110 parses as 0|10|111|0. The shorter average comes from making common symbols cheaper, while still allowing the decoder to find the boundaries.

Distinct descriptions alone do not guarantee this. The assignments $a \mapsto 0$ and $b \mapsto 00$ distinguish isolated symbols if a boundary is supplied, but the string 00 could mean b or aa . A boundary or a separate length announcement is itself additional information; we cannot silently provide it for free when analyzing a stream of descriptions.

Definition 2.1 (Binary prefix code). A binary prefix code on a finite nonempty alphabet $\mathcal{X}$ is a map $c : \mathcal{X} \rightarrow \{0, 1\}^*$ such that, for distinct x, x' , $c(x)$ is not a prefix of $c(x')$. A word counts as its own prefix, so this condition also requires different symbols to have different codewords. Write $\ell(x) = |c(x)|$. For $X \sim p$, its expected length is

$$\mathbb{E}[\ell(X)] = \sum_x p(x)\ell(x).$$

The decoder reads until it reaches a complete codeword, outputs the symbol, and starts again. No codeword can be mistaken for the start of a longer one. In a binary tree, codewords are leaves and bits select left or right branches.

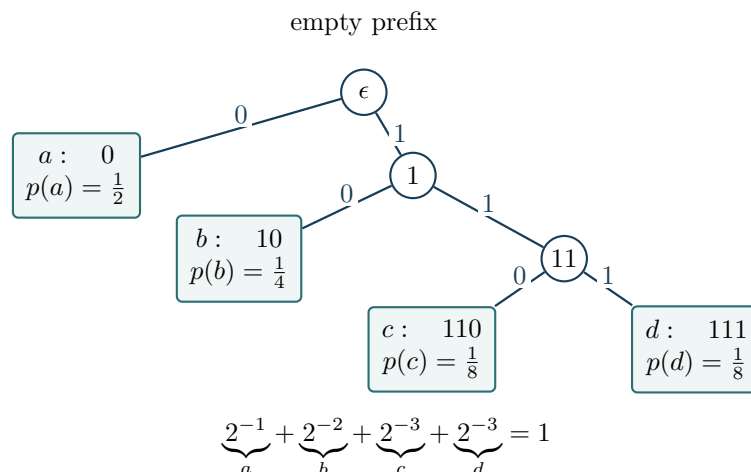


Figure 1: Read a codeword along a path from the root to a leaf. No leaf is an ancestor of another, so no codeword is a prefix of another. The four leaves exhaust Kraft's budget and give expected length $7/4$ bits.

We code only the support of p , and require zero error for every symbol in that support. The distribution and codebook are known to encoder and decoder; their description cost is not included. A singleton support can use the empty word: there is no uncertainty to communicate. This convention presumes that the number of source symbols or blocks is known. Empty words do not convey the length of an otherwise unknown sequence of identical symbols.

The expected-length objective is important. A rare outcome may have a long description. We are not requiring every outcome to fit in $H(X)$ bits.

2.2 Kraft's inequality: which length assignments are possible?

A short codeword excludes many possible longer words. The word 0 rules out every word beginning with 0; it consumes half of the binary tree. The word 10 consumes a quarter. A word at depth ℓ fixes the first ℓ bits and therefore occupies a fraction $2^{-\ell}$ of the nodes at any common depth at least ℓ . The constraint on code lengths is therefore a constraint on how much of the tree the code uses.

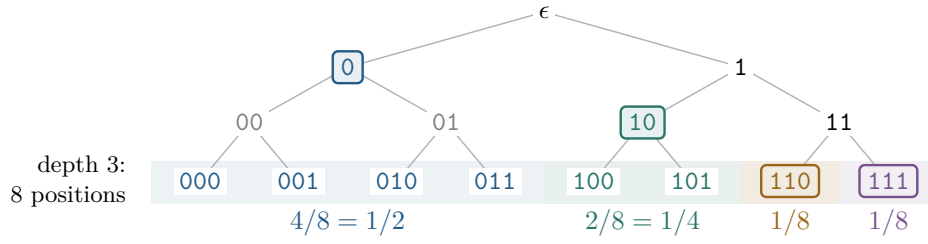
Theorem 2.2 (Kraft's inequality). *Every binary prefix-free code satisfies*

$$\sum_x 2^{-\ell(x)} \leq 1, \tag{2.1}$$

where $\ell(x)$ is the length of the codeword for x .

Converse. *Any assignment of nonnegative integer lengths to a finite nonempty alphabet satisfying this inequality can be achieved by a binary prefix-free code.*

(a) Count disjoint descendants at depth three



(b) Build the code: place shorter words first

Shaded = reserved; white = available; outline = new choice.

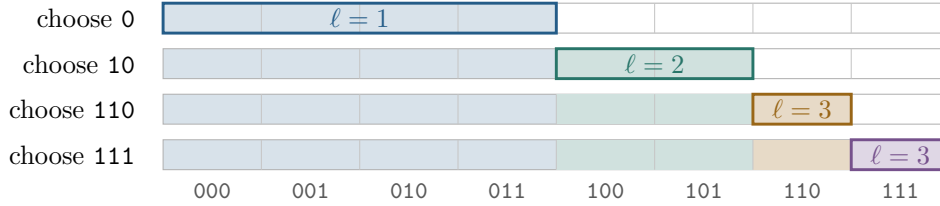


Figure 2: Two views of Kraft's proof for lengths $(1, 2, 3, 3)$. Above, extend every codeword to depth three and count the disjoint sets of descendants: $4 + 2 + 1 + 1 = 8$. Below, choose codewords from shortest to longest; each outlined block is the new choice. All rows show the same eight depth-three positions. The fractions measure tree space, not source probabilities.

Proof. The idea is to count space at a common depth. For the converse direction, choose short words first and show that there is always room for the next word. Figure 2 illustrates both arguments.

Necessity. Enumerate the codewords by $j = 1, \dots, m$ and let $L = \max_j \ell_j$. There are 2^L binary strings of length L . Codeword j is the prefix of exactly $2^{L-\ell_j}$ of them: its first ℓ_j bits are fixed, and its remaining $L - \ell_j$ bits are free. These sets are disjoint. Indeed, if two codewords were both prefixes of the same string, the shorter would be a prefix of the longer. Thus

$$\sum_j 2^{L-\ell_j} \leq 2^L.$$

Dividing by 2^L gives (2.1).

Sufficiency. Sort the requested lengths so that $\ell_1 \leq \dots \leq \ell_m$. Suppose we have already chosen prefix-free words $c_1, \dots, c_{j-1}$ with the required lengths. At depth ℓ_j , word c_i forbids its $2^{\ell_j-\ell_i}$ descendants. Because the earlier words are prefix-free, these forbidden sets are disjoint. Their total size is

$$B_j = \sum_{i < j} 2^{\ell_j-\ell_i}.$$

Dropping the terms with $i > j$ from the assumed Kraft inequality still leaves a sum at most one. Multiplying by 2^{ℓ_j} then gives

$$\sum_{i < j} 2^{-\ell_i} + 2^{-\ell_j} \leq 1 \implies B_j + 1 \leq 2^{\ell_j}.$$

Thus $B_j \leq 2^{\ell_j} - 1$: among the 2^{ℓ_j} candidate nodes, at least one is not forbidden. Choose it for c_j . It is not below an earlier word, and it cannot be above one because all earlier words are at smaller or equal depths. The prefix-free property is preserved, completing the induction. $\square$

Choosing short words first prevents fragmented space: at depth ℓ_j , each earlier word excludes a whole number of candidate nodes. The unused space is therefore a union of complete nodes at that depth. Kraft's bound leaves at least one such node for the next word, exactly as in the lower panel of Figure 2.

Example 2.3 (Following the construction). For lengths $(1, 2, 3, 3)$, choose 0 at depth one. At depth two it forbids 00, 01, so choose 10. At depth three, 0 forbids four words and 10 forbids two more, leaving 110, 111. These give the last two words. The requested lengths have Kraft sum $1/2 + 1/4 + 1/8 + 1/8 = 1$. In contrast, lengths $(1, 1, 2)$ have sum $5/4$ and are impossible: the two one-bit words already exhaust the tree. Lengths $(2, 2)$ are possible and leave half the tree unused, for example with words 00, 01.

Kraft's inequality characterizes *length lists*, not arbitrary words with those lengths. The list $(1, 2)$ is feasible, but the particular words 0, 00 are not prefix-free; 0, 10 realize the same lengths correctly. If one requested length is zero, Kraft forces it to be the only codeword.

The quantities $2^{-\ell(x)}$ act like probability masses whose sum is at most one. This is the bridge from a combinatorial decoding requirement to entropy: Kraft limits the total mass, and Jensen's inequality will turn that limit into a bound on average description length.

2.3 Source coding: entropy and average description length

If fractional bit lengths were possible, the natural assignment would be $\ell(x) = \log(1/p(x))$: its Kraft masses would be exactly $p(x)$, and its average would be $H(p)$. Real lengths are integers. Rounding up should cost less than one bit; proving that no other allocation does better requires a lower bound covering *every* prefix code.

Theorem 2.4 (One-shot lossless source coding). *For every finite source $X \sim p$, every binary prefix code on $\text{supp } p$ satisfies $\mathbb{E}[\ell(X)] \geq H(p)$. There exists such a code with*

$$\boxed{H(p) \leq \mathbb{E}[\ell(X)] < H(p) + 1.} \quad (2.2)$$

Proof. For the lower bound, compare the code lengths with the ideal surprises and use Jensen to turn Kraft's constraint into an average-length bound. For the upper bound, round the ideal lengths and use Kraft's constructive direction.

Lower bound. Put $S = \text{supp } p$. Given a prefix code, Kraft gives

$$K = \sum_{x \in S} 2^{-\ell(x)} \leq 1.$$

The quantity $H(p) - \mathbb{E}[\ell(X)]$ is the average saving compared with the ideal lengths $\log(1/p(x))$. Write this difference as an average of logarithms so that we can apply Jensen:

$$\begin{aligned} H(p) - \mathbb{E}[\ell(X)] &= \sum_{x \in S} p(x) \left(\log \frac{1}{p(x)} - \ell(x) \right) \\ &= \sum_{x \in S} p(x) \log \frac{2^{-\ell(x)}}{p(x)} \\ &\leq \log \left(\sum_{x \in S} p(x) \frac{2^{-\ell(x)}}{p(x)} \right) = \log K \leq 0. \end{aligned}$$

The second equality uses $-\ell(x) = \log 2^{-\ell(x)}$. Jensen's inequality says that an average of logarithms is at most the logarithm of the average; here the weights are $p(x)$ and the positive values being averaged are $2^{-\ell(x)}/p(x)$. Inside the last logarithm, the factor $p(x)$ cancels, leaving precisely the Kraft sum K . Finally $K \leq 1$ implies $\log K \leq 0$. Thus no prefix code can have average length below $H(p)$.

Upper bound. Set the Shannon lengths $\ell(x) = \lceil \log(1/p(x)) \rceil$ for $x \in S$. Since $\ell(x) \geq \log(1/p(x))$, we have $2^{-\ell(x)} \leq p(x)$. Consequently

$$\sum_x 2^{-\ell(x)} \leq \sum_x p(x) = 1.$$

Kraft supplies a code with these lengths. For every x in the support,

$$\log \frac{1}{p(x)} \leq \ell(x) < \log \frac{1}{p(x)} + 1.$$

Multiply by $p(x)$ and sum over S . The left side averages to $H(p)$; the extra 1 on the right averages to $\sum_{x \in S} p(x) = 1$. This proves $H(p) \leq \mathbb{E}[\ell(X)] < H(p) + 1$. $\square$

Equality in the lower bound requires both $K = 1$ and equality in Jensen. Strict concavity says that the ratios $2^{-\ell(x)}/p(x)$ must be constant on S . If their common value is a , then $K = \sum_{x \in S} ap(x) = a$, so $K = 1$ forces $a = 1$. Hence $\mathbb{E}[\ell(X)] = H(p)$ exactly when $p(x) = 2^{-\ell(x)}$ for every supported symbol. Our four-symbol code meets this condition, so its average of 7/4 bits is optimal.

Rounding is a guarantee, not necessarily an optimal algorithm. For three equally likely symbols, Shannon assigns lengths (2, 2, 2) and average 2. The prefix code 0, 10, 11 instead has average 5/3, while entropy is $\log 3 \approx 1.585$. The theorem leaves room between entropy and the best integer solution. One need not solve that optimization exactly to approach entropy per symbol: encode a whole block at once.

Why block coding removes the rounding loss. For n independent copies $X^n = (X_1, \dots, X_n)$ of X , we can treat each entire block x^n as one source symbol and assign it one codeword. Let $\ell_n^*(x^n)$ be the length assigned by a prefix code that minimizes expected block length. The block length n and distribution are known to both parties. Every block of positive probability must be decoded correctly.

Corollary 2.5 (Zero-error, expected-length source coding). *For every $n \geq 1$,*

$$H(X) \leq \frac{\mathbb{E}[\ell_n^*(X^n)]}{n} < H(X) + \frac{1}{n}, \quad \lim_{n \rightarrow \infty} \frac{\mathbb{E}[\ell_n^*(X^n)]}{n} = H(X).$$

Proof. The entropy chain rule and independence give

$$H(X^n) = \sum_{i=1}^n H(X_i | X_{<i}) = \sum_{i=1}^n H(X_i) = nH(X).$$

The second equality uses independence; the last uses the fact that every X_i has the same distribution as X . Apply Theorem 2.4 to the block as a single source symbol. Every block code has expected length at least $nH(X)$, and some block code has expected length less than $nH(X) + 1$. The optimal code lies between these bounds. Divide by n ; since $1/n \rightarrow 0$, the two bounds also prove the limit.

For completeness, an optimal code exists. Fix the expected length C of any feasible code. In any code with expected length at most C , each supported block satisfies $\Pr[X^n = x^n] \ell(x^n) \leq C$. Its positive probability therefore bounds its integer length. There are finitely many such blocks and hence finitely many length lists with cost at most C ; a minimum is attained among them. $\square$

The rounding loss is less than one bit *per block*, rather than per source symbol. For Bernoulli(0.1) bits, coding one bit at a time costs one bit per symbol, although entropy is approximately 0.4690. Coding pairs already improves this:

Source pair	00	01	10	11
Probability	0.81	0.09	0.09	0.01
Codeword	0	10	110	111

Its expected length is $0.81 \cdot 1 + 0.09 \cdot 2 + 0.09 \cdot 3 + 0.01 \cdot 3 = 1.29$ bits per pair, or 0.645 per source bit. For blocks of 100 bits the theorem guarantees an average below 0.4790 bits per source bit. It is an existence statement and does not promise an efficient method for constructing or storing the exponentially large block codebook.

Two different compression questions. The current theorem has zero error and variable length. A later theorem will allow a small error probability while requiring a *fixed* length for every block. These are different resource guarantees. For example, all 2^n binary strings have positive probability under Bernoulli(0.1), so a fixed-length, zero-error encoding using b bits needs $2^b \geq 2^n$ distinct codewords, hence $b \geq n$. A small average length is possible because rare blocks are allowed long descriptions. No concentration or typical-set argument was used here.

2.4 Relative entropy: the cost of a wrong model

The source coding theorem says that the true distribution p determines the best achievable average rate. What if the encoder uses the wrong distribution? This question leads to a measure of mismatch that is also useful for prediction and inference.

Under a model q , outcome x has predicted surprise $\log(1/q(x))$. If the data actually follow p , the average surprise assigned by this model is $\mathbb{E}_p[\log(1/q(X))]$, also called its *cross entropy* or expected log loss. Notice the roles: q assigns the scores, but p weights how often we pay them.

Definition 2.6 (Kullback–Leibler divergence). For distributions p, q on the same finite alphabet, their relative entropy is

$$D(p\|q) = \sum_{x:p(x)>0} p(x) \log \frac{p(x)}{q(x)}.$$

If $p(x) > 0 = q(x)$ for some x , define $D(p\|q) = +\infty$. Terms with $p(x) = 0$ contribute zero.

For every x with $p(x) > 0$ and $q(x) > 0$,

$$\log \frac{1}{q(x)} = \log \frac{1}{p(x)} + \log \frac{p(x)}{q(x)}.$$

Average with weights $p(x)$. The first term on the right averages to entropy and the second to divergence, giving the exact identity

$$\mathbb{E}_p \left[\log \frac{1}{q(X)} \right] = H(p) + D(p\|q). \quad (2.3)$$

Thus divergence is the excess expected log loss from using q instead of the true distribution p . If $q(x) = 0$ for an outcome with $p(x) > 0$, both the cross entropy and divergence are infinite, and the identity still holds in that sense. An event declared impossible by q has infinite cost if it can actually occur. Individual log ratios can be negative: an outcome to which q assigns too much mass is cheap under q . The next theorem says that, averaged under the true distribution, those savings cannot win.

Theorem 2.7 (Gibbs' inequality). $D(p\|q) \geq 0$, with equality if and only if $p = q$.

Proof. The idea is that a probability model has only one unit of total mass to distribute. Jensen's inequality converts that constraint into an average logarithmic penalty. Infinite divergence already satisfies the claim, so assume $q(x) > 0$ whenever $p(x) > 0$ and put $S = \text{supp } p$.

Apply the concavity of $\log$ with weights $p(x)$, which sum to one on S , and positive values $q(x)/p(x)$. Write $q(S) = \sum_{x \in S} q(x)$. Then

$$\begin{aligned} -D(p\|q) &= \sum_{x \in S} p(x) \log \frac{q(x)}{p(x)} \\ &\leq \log \left(\sum_{x \in S} p(x) \frac{q(x)}{p(x)} \right) = \log q(S) \leq 0. \end{aligned}$$

Inside the logarithm, $p(x)$ cancels and the sum becomes $q(S)$. Because q is a probability distribution, $q(S) \leq 1$ and hence $\log q(S) \leq 0$. This proves $-D(p\|q) \leq 0$, or $D(p\|q) \geq 0$. We used an inequality for $q(S)$ because q may place mass outside S ; the two supports need not agree. Equality requires both $q(S) = 1$ and equality in the strictly concave logarithm. The latter says that $q(x)/p(x)$ is a constant a on S . Summing $q(x) = ap(x)$ over S gives $q(S) = a$, so $q(S) = 1$ forces $a = 1$. There is then no mass outside S , and $p = q$ on the whole alphabet. The converse follows immediately from the definition. $\square$

For example, let $p = (0.9, 0.1)$ and $q = (1/2, 1/2)$. The fair model assigns one bit to every outcome, whereas $H(p) = h_2(0.1) \approx 0.4690$ bits. Hence $D(p\|q) \approx 0.5310$ bits. Reversing the roles gives $D(q\|p) = \frac{1}{2} \log \frac{1/2}{0.9} + \frac{1}{2} \log \frac{1/2}{0.1} \approx 0.7370$ bits. Divergence is not a symmetric distance; its direction records which distribution generates the data and which is the model.

2.5 Mismatch, independence, and information gain

Coding with a mismatched distribution. Suppose the data follow p but we design a code using a distribution q , with $q(x) > 0$ whenever $p(x) > 0$. Use Shannon lengths $\ell_q(x) = \lceil \log(1/q(x)) \rceil$. These are feasible because $\sum_{x \in \text{supp } p} 2^{-\ell_q(x)} \leq \sum_{x \in \text{supp } p} q(x) \leq 1$, so Kraft's converse supplies a prefix code.

Rounding gives, for every supported x ,

$$\log \frac{1}{q(x)} \leq \ell_q(x) < \log \frac{1}{q(x)} + 1.$$

Take expectation under the *true* distribution p and use the cross-entropy identity (2.3):

$$H(p) + D(p\|q) \leq \mathbb{E}_p[\ell_q(X)] < H(p) + D(p\|q) + 1. \quad (2.4)$$

Thus $D(p\|q)$ is the exact penalty for ideal log lengths. For integer codewords there is also a rounding cost of less than one bit, so divergence need not equal the difference in expected length between two optimal integer codes.

For independent data $X_1, \dots, X_n \sim p$, apply the same construction to a whole block using the product model $q^{\otimes n}(x^n) = \prod_{i=1}^n q(x_i)$. The block length is

$$\ell_{q,n}(x^n) = \left\lceil \log \frac{1}{q^{\otimes n}(x^n)} \right\rceil = \left\lceil \sum_{i=1}^n \log \frac{1}{q(x_i)} \right\rceil.$$

We round *once for the entire block*, not once for each coordinate. By linearity of expectation and (2.3), the sum inside the ceiling has expectation $n[H(p) + D(p\|q)]$. Rounding adds less than one bit to that block length. Dividing by n gives

$$H(p) + D(p\|q) \leq \frac{1}{n} \mathbb{E}[\ell_{q,n}(X^n)] < H(p) + D(p\|q) + \frac{1}{n}.$$

Rounding vanishes per symbol; a fixed mismatch penalty does not.

For the four-symbol source, the uniform model $q = (1/4, 1/4, 1/4, 1/4)$ uses two bits per outcome. Its excess cost is exactly $2 - 7/4 = 1/4$ bit, which is $D(p\|q)$. Here both sets of ideal lengths happen to be integral.

Revisiting the coding lower bound. Now that divergence is available, we can say exactly where an arbitrary prefix code loses efficiency. Let

$$K = \sum_{x \in \text{supp } p} 2^{-\ell(x)} \leq 1, \quad q_c(x) = \frac{2^{-\ell(x)}}{K}.$$

Here $K > 0$, and q_c is a distribution on $\text{supp } p$ because its values sum to one. It is the probability model suggested by the code lengths: short codewords correspond to large probabilities. Since $\log q_c(x) = -\ell(x) - \log K$, substitution into the definition gives

$$\begin{aligned} D(p\|q_c) &= \sum_{x \in \text{supp } p} p(x)(\log p(x) + \ell(x) + \log K) \\ &= -H(p) + \mathbb{E}[\ell(X)] + \log K. \end{aligned}$$

The three terms are respectively negative entropy, expected code length, and the constant $\log K$ multiplied by $\sum_x p(x) = 1$. Rearranging gives the exact decomposition

$$\boxed{\mathbb{E}[\ell(X)] - H(p) = D(p\|q_c) + \log \frac{1}{K}.} \quad (2.5)$$

The first term is the mismatch between the true source p and the model q_c induced by the code. The second is positive when the code leaves some of the tree unused, meaning $K < 1$. Gibbs' inequality makes the first term nonnegative, and Kraft makes the second nonnegative. This both recovers the earlier lower bound and explains its two possible sources of slack.

Mutual information is departure from independence. Suppose we describe a pair (X, Y) using its separate marginal distributions, ignoring their dependence. This uses the model $(p_X p_Y)(x, y) = p_X(x)p_Y(y)$ in place of the true joint distribution p_{XY} . If $p_{XY}(x, y) > 0$, both marginals are positive, so every ratio below is well defined. Expand the logarithm in the divergence:

$$\begin{aligned} D(p_{XY}\|p_X p_Y) &= \mathbb{E} \left[\log \frac{p_{XY}(X, Y)}{p_X(X)p_Y(Y)} \right] \\ &= \mathbb{E}[\log p_{XY}(X, Y)] - \mathbb{E}[\log p_X(X)] - \mathbb{E}[\log p_Y(Y)] \\ &= H(X) + H(Y) - H(X, Y) = I(X; Y). \end{aligned} \quad (2.6)$$

The three expectations are $-H(X, Y)$, $-H(X)$, and $-H(Y)$, by the definition of entropy. The last equality is the entropy identity for mutual information from Lecture 1. Mutual information therefore measures the log-loss penalty for ignoring dependence. If X is a fair bit and $Y = X$, the joint law gives probabilities $1/2, 1/2$ to $00, 11$, while the product law gives $1/4$ to all four pairs. On either possible outcome the log ratio is one, so $I(X; Y) = 1$ bit.

The posterior view. Before observing Y , our distribution for X is the prior p_X . After seeing $Y = y$, it is the posterior $p_{X|Y=y}$. To express information in terms of this change, factor $p_{XY}(x, y) = p_Y(y)p_{X|Y=y}(x)$ in (2.6). The factor $p_Y(y)$ cancels inside the logarithm. Grouping the expectation by the value of Y gives

$$I(X; Y) = \sum_{y: p_Y(y) > 0} p_Y(y) \sum_{x: p(x|y) > 0} p(x | y) \log \frac{p(x | y)}{p_X(x)}.$$

Here $p(x | y)$ abbreviates $p_{X|Y=y}(x)$. For a fixed y , the inner sum is exactly the divergence of the posterior from the prior:

$$\boxed{I(X; Y) = \sum_{y: p_Y(y) > 0} p_Y(y) D(p_{X|Y=y} \| p_X).} \quad (2.7)$$

Information is therefore the *average* divergence of the posterior from the prior, with each observation weighted by its actual probability. Equivalently, prediction without Y has expected log loss $H(X)$, while prediction using the posterior has expected log loss $H(X | Y)$. Their

difference is $I(X; Y)$. For a particular y , $D(p_{X|y} \| p_X)$ is not generally $H(X) - H(X | Y = y)$: the former is nonnegative, while the latter can be negative. The two descriptions agree after averaging over Y .

For conditional mutual information, fix a value z with $p_Z(z) > 0$. Apply (2.6) to the conditional joint law $p_{XY|z}$ and its marginals $p_{X|z}, p_{Y|z}$, then average over Z :

$$I(X; Y | Z) = \sum_{z: p_Z(z) > 0} p_Z(z) D(p_{XY|z} \| p_{X|z} p_{Y|z}).$$

Each summand is nonnegative by Gibbs' inequality. Their weighted average is zero exactly when every term with positive weight is zero, namely when $p_{XY|z} = p_{X|z} p_{Y|z}$ for each such z . This is precisely conditional independence of X and Y given Z .

2.6 Counting a set by making a random choice from it

Entropy also turns local constraints into bounds on how many objects exist. The useful move is to introduce randomness ourselves: if S is a nonempty finite set, choose an object uniformly from S . Then the unknown quantity $\log |S|$ becomes an entropy that can be decomposed.

Suppose $S \subseteq \mathcal{A}^n$ is nonempty and X^n is uniform on S . Every possible value then has surprise $\log |S|$, so $H(X^n) = \log |S|$. Expand by the chain rule and apply conditioning-reduces-entropy to each term:

$$\log |S| = H(X^n) = \sum_{i=1}^n H(X_i | X_{<i}) \leq \sum_{i=1}^n H(X_i). \quad (2.8)$$

The last step is $H(X_i | X_{<i}) \leq H(X_i)$ for each i . The coordinates need not be independent; that inequality is valid even when they are strongly dependent. It is useful because a constraint on S may make the distribution of a single coordinate easy to bound, even when the size of S itself is unknown.

Example 2.8 (Binary strings of a prescribed weight). Let S contain the length- n binary strings with exactly k ones, where $n \geq 1$ and $0 \leq k \leq n$. Then $|S| = \binom{n}{k}$. For a uniform choice from S , coordinate permutations preserve the distribution, so every coordinate has the same probability, say r , of being one. Since $\sum_i X_i = k$ for every string in S ,

$$nr = \sum_{i=1}^n \Pr[X_i = 1] = \mathbb{E} \left[\sum_{i=1}^n X_i \right] = k.$$

Thus $r = k/n$, and each coordinate is a Bernoulli(k/n) variable with entropy $h_2(k/n)$. Substitution into (2.8) gives $\log \binom{n}{k} \leq nh_2(k/n)$. Exponentiating both sides with base two yields

$$\boxed{\binom{n}{k} \leq 2^{nh_2(k/n)}}. \quad (2.9)$$

The cases $k = 0, n$ give equality $1 = 1$. For $n = 100, k = 10$, the upper bound is $2^{100h_2(0.1)} \approx 2^{46.90}$, compared with 2^{100} unconstrained strings. This exponential saving follows from one-coordinate distributions, without estimating any factorials.

Additional application: compress without knowing the bias. This counting bound suggests a code for an arbitrary binary string x^n . Assume n is known. First send its number k of ones in $\lceil \log(n+1) \rceil$ bits, then its position in a fixed ordering of the weight- k strings in $\lceil \log \binom{n}{k} \rceil$ bits. Both parties know the ordering. The decoder first reads the fixed-size header to learn k , then knows exactly how many payload bits to read and which list to use. Two codewords with

different k have different headers of the same length; two with the same k have the same total length. Thus neither can be a prefix of the other, and the complete code is prefix-free.

The length for a string of weight k is

$$\begin{aligned}\ell(x^n) &= \lceil \log(n+1) \rceil + \left\lceil \log \binom{n}{k} \right\rceil \\ &< \lceil \log(n+1) \rceil + \log \binom{n}{k} + 1 \\ &\leq \lceil \log(n+1) \rceil + nh_2(k/n) + 1.\end{aligned}$$

The strict inequality uses $\lceil a \rceil < a + 1$; the last uses the binomial bound (2.9).

Now suppose $X_1, \dots, X_n$ are independent Bernoulli(p) bits and let $K = \sum_i X_i$ be their random number of ones. Taking expectation in the pointwise length bound gives

$$\mathbb{E}[\ell(X^n)] < \lceil \log(n+1) \rceil + n\mathbb{E}[h_2(K/n)] + 1.$$

The empirical fraction K/n has mean $\mathbb{E}[K/n] = (1/n) \sum_i \mathbb{E}[X_i] = p$. Since binary entropy is concave, Jensen's inequality now gives

$$\mathbb{E}[h_2(K/n)] \leq h_2(\mathbb{E}[K/n]) = h_2(p).$$

This yields the upper bound below. The lower bound follows from source coding, since this is a prefix code and $H(X^n) = nh_2(p)$:

$$h_2(p) \leq \frac{\mathbb{E}[\ell(X^n)]}{n} < h_2(p) + \frac{\lceil \log(n+1) \rceil + 1}{n}.$$

The final fraction tends to zero. Hence this *same code*, which never uses p , has expected rate approaching $h_2(p)$ for every $p \in [0, 1]$. It first describes the empirical count and then specifies the remaining choice within that count. This is an example of universal compression; we make no efficiency claim about finding a string's rank or recovering the string from its rank.

Short exercises and checks

1. Construct a prefix code with lengths $(2, 2, 3, 3, 3, 3)$. Explain at which step the greedy construction would fail for $(1, 2, 2, 3)$.
2. For a uniform distribution on three symbols, compare entropy, Shannon rounding, and the code 0, 10, 11. Why does coding a block resolve the apparent gap between entropy and integral lengths?
3. Let $p = (1/2, 1/4, 1/8, 1/8)$. Compute the average Shannon-code length when the coding model is uniform, and identify both terms on the right side of (2.5) for that code.
4. Choose a string uniformly from $\{001, 010, 100\}$. Compute its joint entropy and the entropy of each coordinate. Compare the two sides of (2.8), and explain why the coordinates are not independent.