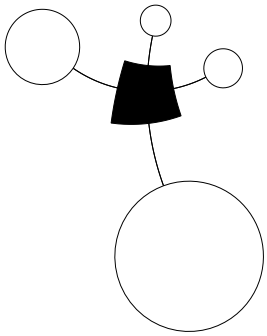
Part I

Data Compression



About Chapter 4

In this chapter we discuss how to measure the information content of the outcome of a random experiment.

This chapter has some tough bits. If you find the mathematical details hard, skim through them and keep going – you'll be able to enjoy Chapters 5 and 6 without this chapter's tools.

Before reading Chapter 4, you should have read Chapter 2 and worked on exercises 2.21–2.25 and 2.16 (pp.36–37), and exercise 4.1 below.

The following exercise is intended to help you think about how to measure information content.

Exercise 4.1.[2, p.69] – *Please work on this problem before reading Chapter 4.*

You are given 12 balls, all equal in weight except for one that is either heavier or lighter. You are also given a two-pan balance to use. In each use of the balance you may put any number of the 12 balls on the left pan, and the same number on the right pan, and push a button to initiate the weighing; there are three possible outcomes: either the weights are equal, or the balls on the left are heavier, or the balls on the left are lighter. Your task is to design a strategy to determine which is the odd ball *and* whether it is heavier or lighter than the others *in as few uses of the balance as possible*.

While thinking about this problem, you may find it helpful to consider the following questions:

(a) How can one measure *information*?

(b) When you have identified the odd ball and whether it is heavy or light, how much information have you gained?

(c) Once you have designed a strategy, draw a tree showing, for each of the possible outcomes of a weighing, what weighing you perform next. At each node in the tree, how much information have the outcomes so far given you, and how much information remains to be gained?

(d) How much information is gained when you learn (i) the state of a flipped coin; (ii) the states of two flipped coins; (iii) the outcome when a four-sided die is rolled?

(e) How much information is gained on the first step of the weighing problem if 6 balls are weighed against the other 6? How much is gained if 4 are weighed against 4 on the first step, leaving out 4 balls?

Notation

$x \in \mathcal{A}$	x is a *member* of the set $\mathcal{A}$
$\mathcal{S} \subset \mathcal{A}$	$\mathcal{S}$ is a *subset* of the set $\mathcal{A}$
$\mathcal{S} \subseteq \mathcal{A}$	$\mathcal{S}$ is a subset of, or equal to, the set $\mathcal{A}$
$\mathcal{V} = \mathcal{B} \cup \mathcal{A}$	$\mathcal{V}$ is the *union* of the sets $\mathcal{B}$ and $\mathcal{A}$
$\mathcal{V} = \mathcal{B} \cap \mathcal{A}$	$\mathcal{V}$ is the *intersection* of the sets $\mathcal{B}$ and $\mathcal{A}$
$\lvert\mathcal{A}\rvert$	number of elements in set $\mathcal{A}$

4

The Source Coding Theorem

▶ 4.1 How to measure the information content of a random variable?

In the next few chapters, we'll be talking about probability distributions and random variables. Most of the time we can get by with sloppy notation, but occasionally, we will need precise notation. Here is the notation that we established in Chapter 2.

An ensemble X is a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$, where the *outcome* x is the value of a random variable, which takes on one of a set of possible values, $\mathcal{A}_X = \{a_1, a_2, \ldots, a_i, \ldots, a_I\}$, having probabilities $\mathcal{P}_X = \{p_1, p_2, \ldots, p_I\}$, with $P(x=a_i) = p_i$, $p_i \geq 0$ and $\sum_{a_i \in \mathcal{A}_X} P(x=a_i) = 1$.

How can we measure the information content of an outcome $x = a_i$ from such an ensemble? In this chapter we examine the assertions

1. that the *Shannon information content*,

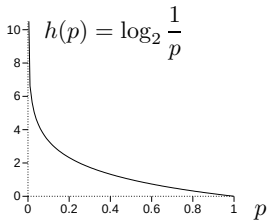
$$h(x=a_i) \equiv \log_2 \frac{1}{p_i}, \tag{4.1}$$

   is a sensible measure of the information content of the outcome $x = a_i$, and

2. that the *entropy* of the ensemble,

$$H(X) = \sum_i p_i \log_2 \frac{1}{p_i}, \tag{4.2}$$

   is a sensible measure of the ensemble's average information content.



p	$h(p)$	$H_2(p)$
0.001	10.0	0.011
0.01	6.6	0.081
0.1	3.3	0.47
0.2	2.3	0.72
0.5	1.0	1.0

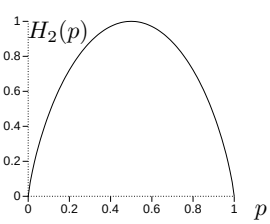


Figure 4.1. The Shannon information content $h(p) = \log_2 \frac{1}{p}$ and the binary entropy function $H_2(p) = H(p, 1-p) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{(1-p)}$ as a function of p.

Figure 4.1 shows the Shannon information content of an outcome with probability p, as a function of p. The less probable an outcome is, the greater its Shannon information content. Figure 4.1 also shows the binary entropy function,

$$H_2(p) = H(p, 1-p) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{(1-p)}, \tag{4.3}$$

which is the entropy of the ensemble X whose alphabet and probability distribution are $\mathcal{A}_X = \{a, b\}, \mathcal{P}_X = \{p, (1-p)\}$.

Information content of independent random variables

Why should $\log 1/p_i$ have anything to do with the information content? Why not some other function of p_i? We'll explore this question in detail shortly, but first, notice a nice property of this particular function $h(x) = \log 1/p(x)$.

Imagine learning the value of two *independent* random variables, x and y. The definition of independence is that the probability distribution is separable into a *product*:

$$P(x, y) = P(x)P(y). \tag{4.4}$$

Intuitively, we might want any measure of the 'amount of information gained' to have the property of *additivity* – that is, for independent random variables x and y, the information gained when we learn x and y should equal the sum of the information gained if x alone were learned and the information gained if y alone were learned.

The Shannon information content of the outcome x, y is

$$h(x, y) = \log \frac{1}{P(x, y)} = \log \frac{1}{P(x)P(y)} = \log \frac{1}{P(x)} + \log \frac{1}{P(y)} \tag{4.5}$$

so it does indeed satisfy

$$h(x, y) = h(x) + h(y), \text{ if } x \text{ and } y \text{ are independent.} \tag{4.6}$$

Exercise 4.2.[1, p.86] Show that, if x and y are independent, the entropy of the outcome x, y satisfies

$$H(X, Y) = H(X) + H(Y). \tag{4.7}$$

In words, entropy is additive for independent variables.

We now explore these ideas with some examples; then, in section 4.4 and in Chapters 5 and 6, we prove that the Shannon information content and the entropy are related to the number of bits needed to describe the outcome of an experiment.

The weighing problem: designing informative experiments

Have you solved the weighing problem (exercise 4.1, p.66) yet? Are you sure? Notice that in three uses of the balance – which reads either 'left heavier', 'right heavier', or 'balanced' – the number of conceivable outcomes is $3^3 = 27$, whereas the number of possible states of the world is 24: the odd ball could be any of twelve balls, and it could be heavy or light. So in principle, the problem might be solvable in three weighings – but not in two, since $3^2 < 24$.

If you know how you can determine the odd weight *and* whether it is heavy or light in *three* weighings, then you may read on. If you haven't found a strategy that always gets there in three weighings, I encourage you to think about exercise 4.1 some more.

Why is your strategy optimal? What is it about your series of weighings that allows useful information to be gained as quickly as possible? The answer is that at each step of an optimal procedure, the three outcomes ('left heavier', 'right heavier', and 'balance') are *as close as possible to equiprobable*. An optimal solution is shown in figure 4.2.

Suboptimal strategies, such as weighing balls 1–6 against 7–12 on the first step, do not achieve all outcomes with equal probability: these two sets of balls can never balance, so the only possible outcomes are 'left heavy' and 'right heavy'. Such a binary outcome only rules out half of the possible hypotheses,

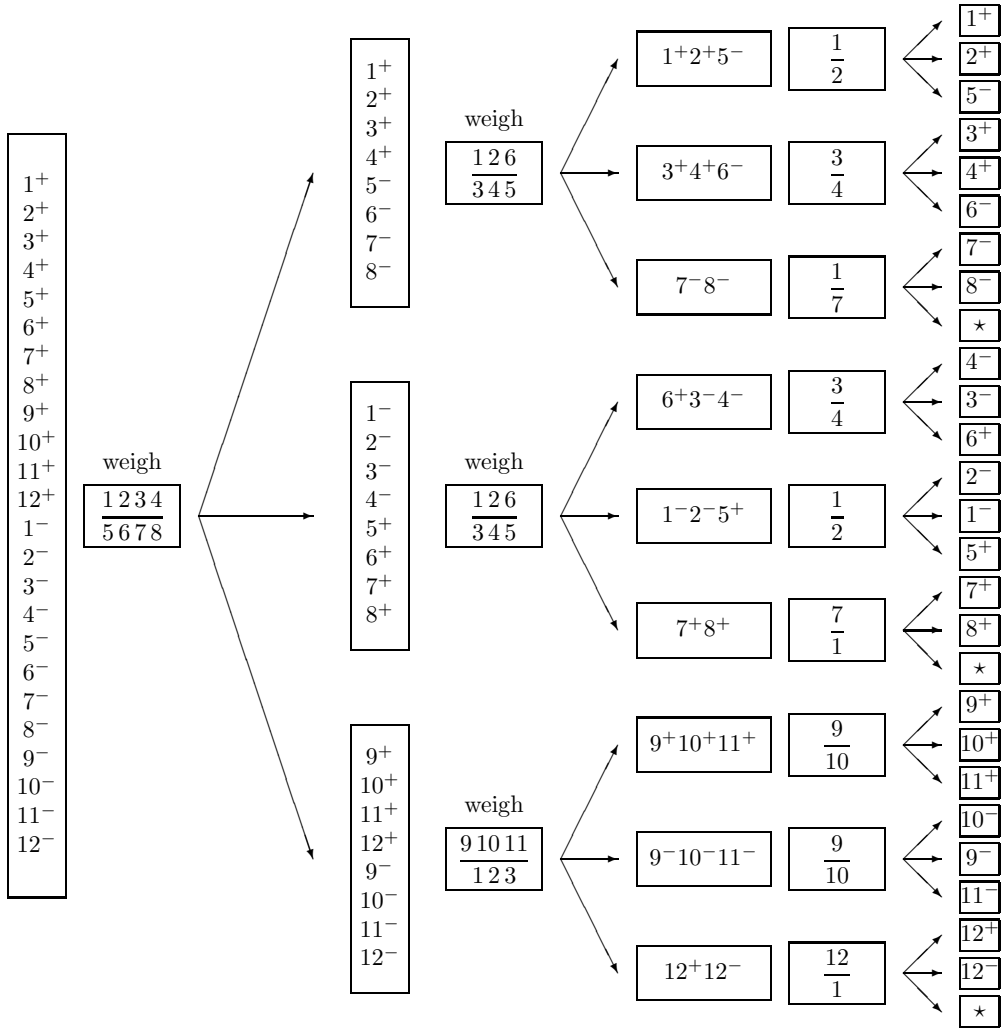


Figure 4.2. An optimal solution to the weighing problem. At each step there are two boxes: the left box shows which hypotheses are still possible; the right box shows the balls involved in the next weighing. The 24 hypotheses are written $1^+, \ldots, 12^-$, with, e.g., 1^+ denoting that 1 is the odd ball and it is heavy. Weighings are written by listing the names of the balls on the two pans, separated by a line; for example, in the first weighing, balls 1, 2, 3, and 4 are put on the left-hand side and 5, 6, 7, and 8 on the right. In each triplet of arrows the upper arrow leads to the situation when the left side is heavier, the middle arrow to the situation when the right side is heavier, and the lower arrow to the situation when the outcome is balanced. The three points labelled $\star$ correspond to impossible outcomes.

so a strategy that uses such outcomes must sometimes take longer to find the right answer.

The insight that the outcomes should be as near as possible to equiprobable makes it easier to search for an optimal strategy. The first weighing must divide the 24 possible hypotheses into three groups of eight. Then the second weighing must be chosen so that there is a 3:3:2 split of the hypotheses.

Thus we might conclude:

> the outcome of a random experiment is guaranteed to be most informative if the probability distribution over outcomes is uniform.

This conclusion agrees with the property of the entropy that you proved when you solved exercise 2.25 (p.37): the entropy of an ensemble X is biggest if all the outcomes have equal probability $p_i = 1/|\mathcal{A}_X|$.

Guessing games

In the game of twenty questions, one player thinks of an object, and the other player attempts to guess what the object is by asking questions that have yes/no answers, for example, 'is it alive?', or 'is it human?' The aim is to identify the object with as few questions as possible. What is the best strategy for playing this game? For simplicity, imagine that we are playing the rather dull version of twenty questions called 'sixty-three'.

Example 4.3. The game 'sixty-three'. What's the smallest number of yes/no questions needed to identify an integer x between 0 and 63?

Intuitively, the best questions successively divide the 64 possibilities into equal sized sets. Six questions suffice. One reasonable strategy asks the following questions:

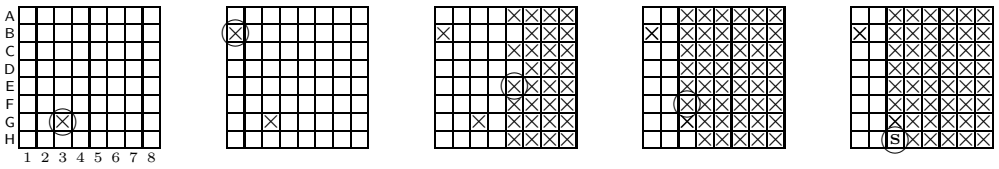
1: is $x \geq 32$?
2: is $x \bmod 32 \geq 16$?
3: is $x \bmod 16 \geq 8$?
4: is $x \bmod 8 \geq 4$?
5: is $x \bmod 4 \geq 2$?
6: is $x \bmod 2 = 1$?

[The notation $x \bmod 32$, pronounced 'x modulo 32', denotes the remainder when x is divided by 32; for example, $35 \bmod 32 = 3$ and $32 \bmod 32 = 0$.]

The answers to these questions, if translated from $\{\text{yes}, \text{no}\}$ to $\{\mathtt{1}, \mathtt{0}\}$, give the binary expansion of x, for example $35 \Rightarrow \mathtt{100011}$. □

What are the Shannon information contents of the outcomes in this example? If we assume that all values of x are equally likely, then the answers to the questions are independent and each has Shannon information content $\log_2(1/0.5) = 1\,\text{bit}$; the total Shannon information gained is always six bits. Furthermore, the number x that we learn from these questions is a six-bit binary number. Our questioning strategy defines a way of encoding the random variable x as a binary file.

So far, the Shannon information content makes sense: it measures the length of a binary file that encodes x. However, we have not yet studied ensembles where the outcomes have unequal probabilities. Does the Shannon information content make sense there too?



move #	1	2	32	48	49
question	G3	B1	E5	F3	H3
outcome	$x = \mathtt{n}$	$x = \mathtt{n}$	$x = \mathtt{n}$	$x = \mathtt{n}$	$x = \mathtt{y}$
$P(x)$	$\frac{63}{64}$	$\frac{62}{63}$	$\frac{32}{33}$	$\frac{16}{17}$	$\frac{1}{16}$
$h(x)$	0.0227	0.0230	0.0443	0.0874	4.0
Total info.	0.0227	0.0458	1.0	2.0	6.0

Figure 4.3. A game of `submarine`. The submarine is hit on the 49th attempt.

The game of submarine: how many bits can one bit convey?

In the game of battleships, each player hides a fleet of ships in a sea represented by a square grid. On each turn, one player attempts to hit the other's ships by firing at one square in the opponent's sea. The response to a selected square such as 'G3' is either 'miss', 'hit', or 'hit and destroyed'.

In a boring version of battleships called `submarine`, each player hides just one submarine in one square of an eight-by-eight grid. Figure 4.3 shows a few pictures of this game in progress: the circle represents the square that is being fired at, and the $\times$s show squares in which the outcome was a miss, $x = \mathtt{n}$; the submarine is hit (outcome $x = \mathtt{y}$ shown by the symbol `s`) on the 49th attempt.

Each shot made by a player defines an ensemble. The two possible outcomes are $\{\mathtt{y}, \mathtt{n}\}$, corresponding to a hit and a miss, and their probabilities depend on the state of the board. At the beginning, $P(\mathtt{y}) = 1/64$ and $P(\mathtt{n}) = 63/64$. At the second shot, if the first shot missed, $P(\mathtt{y}) = 1/63$ and $P(\mathtt{n}) = 62/63$. At the third shot, if the first two shots missed, $P(\mathtt{y}) = 1/62$ and $P(\mathtt{n}) = 61/62$.

The Shannon information gained from an outcome x is $h(x) = \log(1/P(x))$. If we are lucky, and hit the submarine on the first shot, then

$$h(x) = h_{(1)}(\mathtt{y}) = \log_2 64 = 6\,\text{bits}. \tag{4.8}$$

Now, it might seem a little strange that one binary outcome can convey six bits. But we have learnt the hiding place, which could have been any of 64 squares; so we have, by one lucky binary question, indeed learnt six bits.

What if the first shot misses? The Shannon information that we gain from this outcome is

$$h(x) = h_{(1)}(\mathtt{n}) = \log_2 \frac{64}{63} = 0.0227\,\text{bits}. \tag{4.9}$$

Does this make sense? It is not so obvious. Let's keep going. If our second shot also misses, the Shannon information content of the second outcome is

$$h_{(2)}(\mathtt{n}) = \log_2 \frac{63}{62} = 0.0230\,\text{bits}. \tag{4.10}$$

If we miss thirty-two times (firing at a new square each time), the total Shannon information gained is

$$\begin{aligned} &\log_2 \frac{64}{63} + \log_2 \frac{63}{62} + \cdots + \log_2 \frac{33}{32} \\ &= 0.0227 + 0.0230 + \cdots + 0.0430 \;=\; 1.0\,\text{bits}. \end{aligned} \tag{4.11}$$

Why this round number? Well, what have we learnt? We now know that the submarine is not in any of the 32 squares we fired at; learning that fact is just like playing a game of `sixty-three` (p.70), asking as our first question 'is x one of the thirty-two numbers corresponding to these squares I fired at?', and receiving the answer 'no'. This answer rules out half of the hypotheses, so it gives us one bit.

After 48 unsuccessful shots, the information gained is 2 bits: the unknown location has been narrowed down to one quarter of the original hypothesis space.

What if we hit the submarine on the 49th shot, when there were 16 squares left? The Shannon information content of this outcome is

$$h_{(49)}(\mathtt{y}) = \log_2 16 = 4.0\,\text{bits}. \tag{4.12}$$

The total Shannon information content of all the outcomes is

$$\begin{aligned} &\log_2 \frac{64}{63} + \log_2 \frac{63}{62} + \cdots + \log_2 \frac{17}{16} + \log_2 \frac{16}{1} \\ &\quad = \quad 0.0227 + 0.0230 + \cdots + 0.0874 + 4.0 \quad = \quad 6.0\,\text{bits}. \end{aligned} \tag{4.13}$$

So once we know where the submarine is, the total Shannon information content gained is 6 bits.

This result holds regardless of when we hit the submarine. If we hit it when there are n squares left to choose from – n was 16 in equation (4.13) – then the total information gained is:

$$\begin{aligned} &\log_2 \frac{64}{63} + \log_2 \frac{63}{62} + \cdots + \log_2 \frac{n+1}{n} + \log_2 \frac{n}{1} \\ &\quad = \quad \log_2 \left[\frac{64}{63} \times \frac{63}{62} \times \cdots \times \frac{n+1}{n} \times \frac{n}{1} \right] \quad = \quad \log_2 \frac{64}{1} \quad = \quad 6\,\text{bits}. \end{aligned} \tag{4.14}$$

What have we learned from the examples so far? I think the `submarine` example makes quite a convincing case for the claim that the Shannon information content is a sensible measure of information content. And the game of `sixty-three` shows that the Shannon information content can be intimately connected to the size of a file that encodes the outcomes of a random experiment, thus suggesting a possible connection to data compression.

In case you're not convinced, let's look at one more example.

The Wenglish language

Wenglish is a language similar to English. Wenglish sentences consist of words drawn at random from the Wenglish dictionary, which contains $2^{15} = 32{,}768$ words, all of length 5 characters. Each word in the Wenglish dictionary was constructed at random by picking five letters from the probability distribution over `a...z` depicted in figure 2.1.

Some entries from the dictionary are shown in alphabetical order in figure 4.4. Notice that the number of words in the dictionary (32,768) is much smaller than the total number of possible words of length 5 letters, $26^5 \simeq 12{,}000{,}000$.

Because the probability of the letter `z` is about 1/1000, only 32 of the words in the dictionary begin with the letter `z`. In contrast, the probability of the letter `a` is about 0.0625, and 2048 of the words begin with the letter `a`. Of those 2048 words, two start `az`, and 128 start `aa`.

Let's imagine that we are reading a Wenglish document, and let's discuss the Shannon information content of the characters as we acquire them. If we

1	`aaail`
2	`aaaiu`
3	`aaald`
	⋮
129	`abati`
	⋮
2047	`azpan`
2048	`aztdn`
	⋮
	⋮
16 384	`odrcr`
	⋮
	⋮
32 737	`zatnt`
	⋮
32 768	`zxast`

Figure 4.4. The Wenglish dictionary.

are given the text one word at a time, the Shannon information content of each five-character word is $\log 32{,}768 = 15$ bits, since Wenglish uses all its words with equal probability. The average information content per character is therefore 3 bits.

Now let's look at the information content if we read the document one character at a time. If, say, the first letter of a word is `a`, the Shannon information content is $\log 1/0.0625 \simeq 4$ bits. If the first letter is `z`, the Shannon information content is $\log 1/0.001 \simeq 10$ bits. The information content is thus highly variable at the first character. The total information content of the 5 characters in a word, however, is exactly 15 bits; so the letters that follow an initial `z` have lower average information content per character than the letters that follow an initial `a`. A rare initial letter such as `z` indeed conveys more information about what the word is than a common initial letter.

Similarly, in English, if rare characters occur at the start of the word (e.g. `xyl...`), then often we can identify the whole word immediately; whereas words that start with common characters (e.g. `pro...`) require more characters before we can identify them.

▶ 4.2 Data compression

The preceding examples justify the idea that the Shannon information content of an outcome is a natural measure of its information content. Improbable outcomes do convey more information than probable outcomes. We now discuss the information content of a source by considering how many bits are needed to describe the outcome of an experiment.

If we can show that we can compress data from a particular source into a file of L bits per source symbol and recover the data reliably, then we will say that the average information content of that source is at most L bits per symbol.

Example: compression of text files

A file is composed of a sequence of bytes. A byte is composed of 8 bits and can have a decimal value between 0 and 255. A typical text file is composed of the ASCII character set (decimal values 0 to 127). This character set uses only seven of the eight bits in a byte.

Here we use the word 'bit' with its meaning, 'a symbol with two values', not to be confused with the unit of information content.

▷ Exercise 4.4.[1, p.86] By how much could the size of a file be reduced given that it is an ASCII file? How would you achieve this reduction?

Intuitively, it seems reasonable to assert that an ASCII file contains 7/8 as much information as an arbitrary file of the same size, since we already know one out of every eight bits before we even look at the file. This is a simple example of redundancy. Most sources of data have further redundancy: English text files use the ASCII characters with non-equal frequency; certain pairs of letters are more probable than others; and entire words can be predicted given the context and a semantic understanding of the text.

Some simple data compression methods that define measures of information content

One way of measuring the information content of a random variable is simply to count the number of *possible* outcomes, $|\mathcal{A}_X|$. (The number of elements in a set $\mathcal{A}$ is denoted by $|\mathcal{A}|$.) If we gave a binary name to each outcome, the

length of each name would be $\log_2 |\mathcal{A}_X|$ bits, if $|\mathcal{A}_X|$ happened to be a power of 2. We thus make the following definition.

The raw bit content of X is

$$H_0(X) = \log_2 |\mathcal{A}_X|. \tag{4.15}$$

$H_0(X)$ is a lower bound for the number of binary questions that are always guaranteed to identify an outcome from the ensemble X. It is an additive quantity: the raw bit content of an ordered pair x, y, having $|\mathcal{A}_X||\mathcal{A}_Y|$ possible outcomes, satisfies

$$H_0(X, Y) = H_0(X) + H_0(Y). \tag{4.16}$$

This measure of information content does not include any probabilistic element, and the encoding rule it corresponds to does not 'compress' the source data, it simply maps each outcome to a constant-length binary string.

Exercise 4.5.[2, p.86] Could there be a compressor that maps an outcome x to a binary code $c(x)$, and a decompressor that maps c back to x, such that *every possible outcome* is compressed into a binary code of length *shorter* than $H_0(X)$ bits?

Even though a simple counting argument shows that it is impossible to make a reversible compression program that reduces the size of *all* files, amateur compression enthusiasts frequently announce that they have invented a program that can do this – indeed that they can further compress compressed files by putting them through their compressor several times. Stranger yet, patents have been granted to these modern-day alchemists. See the `comp.compression` frequently asked questions for further reading.[1]

There are only two ways in which a 'compressor' can actually compress files:

1. A *lossy* compressor compresses some files, but maps some files to the *same* encoding. We'll assume that the user requires perfect recovery of the source file, so the occurrence of one of these confusable files leads to a failure (though in applications such as image compression, lossy compression is viewed as satisfactory). We'll denote by δ the probability that the source string is one of the confusable files, so a lossy compressor has a probability δ of failure. If δ can be made very small then a lossy compressor may be practically useful.

2. A *lossless* compressor maps all files to different encodings; if it shortens some files, it necessarily *makes others longer*. We try to design the compressor so that the probability that a file is lengthened is very small, and the probability that it is shortened is large.

In this chapter we discuss a simple lossy compressor. In subsequent chapters we discuss lossless compression methods.

▶ 4.3 Information content defined in terms of lossy compression

Whichever type of compressor we construct, we need somehow to take into account the *probabilities* of the different outcomes. Imagine comparing the information contents of two text files – one in which all 128 ASCII characters

[1] `http://sunsite.org.uk/public/usenet/news-faqs/comp.compression/`

are used with equal probability, and one in which the characters are used with their frequencies in English text. Can we define a measure of information content that distinguishes between these two files? Intuitively, the latter file contains less information per character because it is more predictable.

One simple way to use our knowledge that some symbols have a smaller probability is to imagine recoding the observations into a smaller alphabet – thus losing the ability to encode some of the more improbable symbols – and then measuring the raw bit content of the new alphabet. For example, we might take a risk when compressing English text, guessing that the most infrequent characters won't occur, and make a reduced ASCII code that omits the characters { !, @, #, %, ^, *, ~, <, >, /, \, _, {, }, [,], | }, thereby reducing the size of the alphabet by seventeen. The larger the risk we are willing to take, the smaller our final alphabet becomes.

We introduce a parameter δ that describes the risk we are taking when using this compression method: δ is the probability that there will be no name for an outcome x.

Example 4.6. Let

$$\begin{aligned} \mathcal{A}_X &= \{ \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h} \}, \\ \text{and} \quad \mathcal{P}_X &= \{ \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{3}{16}, \tfrac{1}{64}, \tfrac{1}{64}, \tfrac{1}{64}, \tfrac{1}{64} \}. \end{aligned} \tag{4.17}$$

The raw bit content of this ensemble is 3 bits, corresponding to 8 binary names. But notice that $P(x \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}) = 15/16$. So if we are willing to run a risk of $\delta = 1/16$ of not having a name for x, then we can get by with four names – half as many names as are needed if every $x \in \mathcal{A}_X$ has a name.

Table 4.5 shows binary names that could be given to the different outcomes in the cases $\delta = 0$ and $\delta = 1/16$. When $\delta = 0$ we need 3 bits to encode the outcome; when $\delta = 1/16$ we only need 2 bits.

$\delta = 0$		$\delta = 1/16$	
x	$c(x)$	x	$c(x)$
a	000	a	00
b	001	b	01
c	010	c	10
d	011	d	11
e	100	e	–
f	101	f	–
g	110	g	–
h	111	h	–

Table 4.5. Binary names for the outcomes, for two failure probabilities δ.

Let us now formalize this idea. To make a compression strategy with risk δ, we make the smallest possible subset S_δ such that the probability that x is not in S_δ is less than or equal to δ, i.e., $P(x \notin S_\delta) \leq \delta$. For each value of δ we can then define a new measure of information content – the log of the size of this smallest subset S_δ. [In ensembles in which several elements have the same probability, there may be several smallest subsets that contain different elements, but all that matters is their sizes (which are equal), so we will not dwell on this ambiguity.]

The smallest δ-sufficient subset S_δ is the smallest subset of $\mathcal{A}_X$ satisfying

$$P(x \in S_\delta) \geq 1 - \delta. \tag{4.18}$$

The subset S_δ can be constructed by ranking the elements of $\mathcal{A}_X$ in order of decreasing probability and adding successive elements starting from the most probable elements until the total probability is $\geq (1-\delta)$.

We can make a data compression code by assigning a binary name to each element of the smallest sufficient subset. This compression scheme motivates the following measure of information content:

The essential bit content of X is:

$$H_\delta(X) = \log_2 |S_\delta| \tag{4.19}$$

Note that $H_0(X)$ is the special case of $H_\delta(X)$ with $\delta = 0$ (if $P(x) > 0$ for all $x \in \mathcal{A}_X$). [Caution: do not confuse $H_0(X)$ and $H_\delta(X)$ with the function $H_2(p)$ displayed in figure 4.1.]

Figure 4.6 shows $H_\delta(X)$ for the ensemble of example 4.6 as a function of δ.

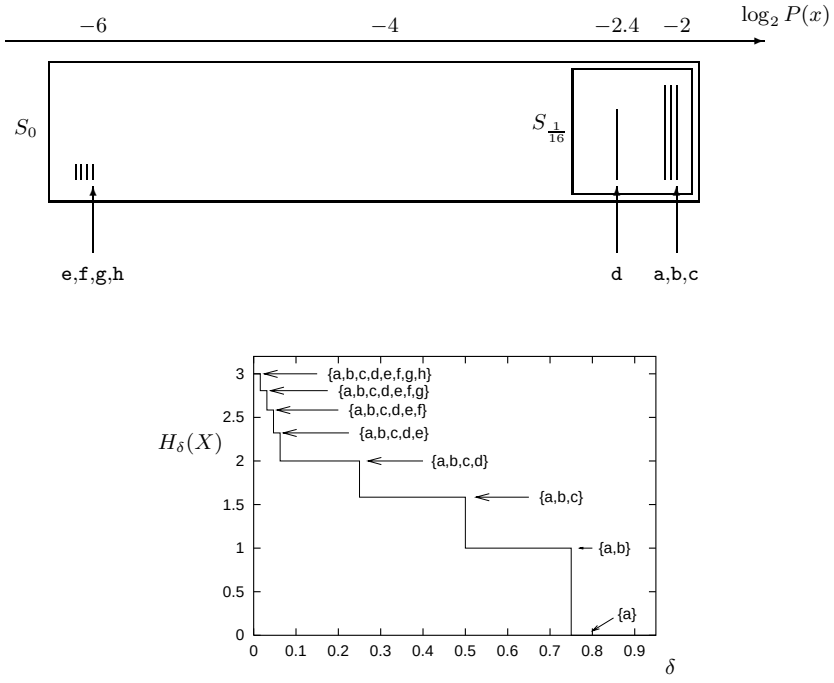


Figure 4.6. (a) The outcomes of X (from example 4.6 (p.75)), ranked by their probability. (b) The essential bit content $H_\delta(X)$. The labels on the graph show the smallest sufficient set as a function of δ. Note $H_0(X) = 3$ bits and $H_{1/16}(X) = 2$ bits.

Extended ensembles

Is this compression method any more useful if we compress *blocks* of symbols from a source?

We now turn to examples where the outcome $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is a string of N independent identically distributed random variables from a single ensemble X. We will denote by X^N the ensemble $(X_1, X_2, \ldots, X_N)$. Remember that entropy is additive for independent variables (exercise 4.2 (p.68)), so $H(X^N) = NH(X)$.

Example 4.7. Consider a string of N flips of a bent coin, $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, where $x_n \in \{0, 1\}$, with probabilities $p_0 = 0.9$, $p_1 = 0.1$. The most probable strings $\mathbf{x}$ are those with most 0s. If $r(\mathbf{x})$ is the number of 1s in $\mathbf{x}$ then

$$P(\mathbf{x}) = p_0^{N-r(\mathbf{x})} p_1^{r(\mathbf{x})}. \tag{4.20}$$

To evaluate $H_\delta(X^N)$ we must find the smallest sufficient subset S_δ. This subset will contain all $\mathbf{x}$ with $r(\mathbf{x}) = 0, 1, 2, \ldots$, up to some $r_{\max}(\delta) - 1$, and some of the $\mathbf{x}$ with $r(\mathbf{x}) = r_{\max}(\delta)$. Figures 4.7 and 4.8 show graphs of $H_\delta(X^N)$ against δ for the cases $N = 4$ and $N = 10$. The steps are the values of δ at which $|S_\delta|$ changes by 1, and the cusps where the slope of the staircase changes are the points where $r_{\max}$ changes by 1.

Exercise 4.8.[2, p.86] What are the mathematical shapes of the curves between the cusps?

For the examples shown in figures 4.6–4.8, $H_\delta(X^N)$ depends strongly on the value of δ, so it might not seem a fundamental or useful definition of information content. But we will consider what happens as N, the number of independent variables in X^N, increases. We will find the remarkable result that $H_\delta(X^N)$ becomes almost independent of δ – and for all δ it is very close to $NH(X)$, where $H(X)$ is the entropy of one of the random variables.

Figure 4.9 illustrates this asymptotic tendency for the binary ensemble of example 4.7. As N increases, $\frac{1}{N} H_\delta(X^N)$ becomes an increasingly flat function,

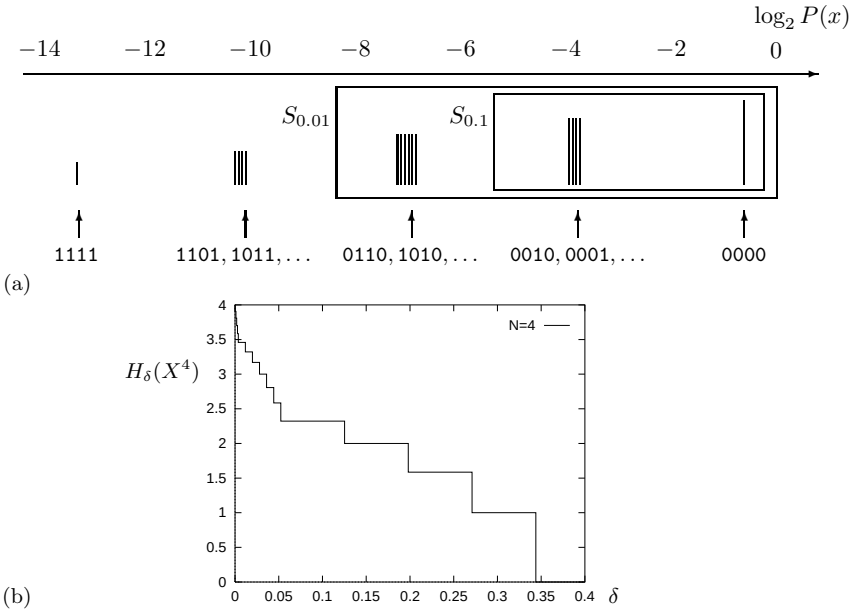


Figure 4.7. (a) The sixteen outcomes of the ensemble X^4 with $p_1 = 0.1$, ranked by probability. (b) The essential bit content $H_\delta(X^4)$. The upper schematic diagram indicates the strings's probabilities by the vertical lines's lengths (not to scale).

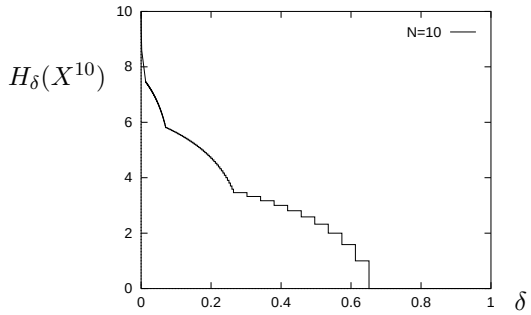


Figure 4.8. $H_\delta(X^N)$ for $N = 10$ binary variables with $p_1 = 0.1$.

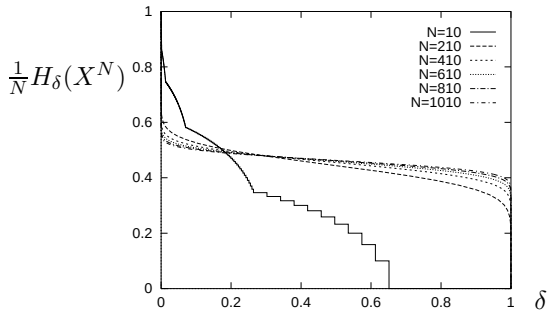


Figure 4.9. $\frac{1}{N}H_\delta(X^N)$ for $N = 10, 210, \ldots, 1010$ binary variables with $p_1 = 0.1$.

$\mathbf{x}$	$\log_2(P(\mathbf{x}))$
...1.....1.....1....1.1.....1.....1.....1.....1.....1.....11...	−50.1
.....1.....1.....1.....1....1.....1.....1.....1....	−37.3
.....1....1..1...1....11..1.1.....11.....1...1.1..1...1.....1.	−65.9
1.1...1.....1.....11.1..1.....1.....1.....1....1..1.11.....	−56.4
...11.....1...1.....1.1.....1.....1....1...1.....1.....1.....	−53.2
.....1.....1.....1.1.....1.....1.....1...1.....1.....1.....	−43.7
.....1.....1.....1...1.....1.....1.....1.....1..11.....	−46.8
.....1..1..1.....111.....1.....1.....1.1...1...1.....1	−56.4
.....1.....1.....1.....1.....1....1.....1....	−37.3
.....1.....1.....1....1..1.1.1.1..1.....1.	−43.7
1.....1.....1...1.....1....1....1.....1..11..1.1...1.....	−56.4
.....11.1.....1.....1.....1.....1.....1.....	−37.3
.1.....1...1.1.....1.....11.....1.1...1.....1.....11.....	−56.4
.....1...1..1.....1..11.1.1.1...1.....1.....1.....1..1.....	−59.5
.....11.1.....1....1..1.....1.....1.....1.....1.....	−46.8
.....	−15.2
11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111	−332.1

Figure 4.10. The top 15 strings are samples from X^{100}, where $p_1 = 0.1$ and $p_0 = 0.9$. The bottom two are the most and least probable strings in this ensemble. The final column shows the log-probabilities of the random strings, which may be compared with the entropy $H(X^{100}) = 46.9$ bits.

except for tails close to $\delta = 0$ and 1. As long as we are allowed a tiny probability of error δ, compression down to NH bits is possible. Even if we are allowed a large probability of error, we still can compress only down to NH bits. This is the source coding theorem.

Theorem 4.1 Shannon's source coding theorem. *Let X be an ensemble with entropy $H(X) = H$ bits. Given $\epsilon > 0$ and $0 < \delta < 1$, there exists a positive integer N_0 such that for $N > N_0$,*

$$\left| \frac{1}{N} H_\delta(X^N) - H \right| < \epsilon. \tag{4.21}$$

4.4 Typicality

Why does increasing N help? Let's examine long strings from X^N. Table 4.10 shows fifteen samples from X^N for $N = 100$ and $p_1 = 0.1$. The probability of a string $\mathbf{x}$ that contains r 1s and $N-r$ 0s is

$$P(\mathbf{x}) = p_1^r (1 - p_1)^{N-r}. \tag{4.22}$$

The number of strings that contain r 1s is

$$n(r) = \binom{N}{r}. \tag{4.23}$$

So the number of 1s, r, has a binomial distribution:

$$P(r) = \binom{N}{r} p_1^r (1 - p_1)^{N-r}. \tag{4.24}$$

These functions are shown in figure 4.11. The mean of r is Np_1, and its standard deviation is $\sqrt{Np_1(1 - p_1)}$ (p.1). If N is 100 then

$$r \sim Np_1 \pm \sqrt{Np_1(1 - p_1)} \simeq 10 \pm 3. \tag{4.25}$$

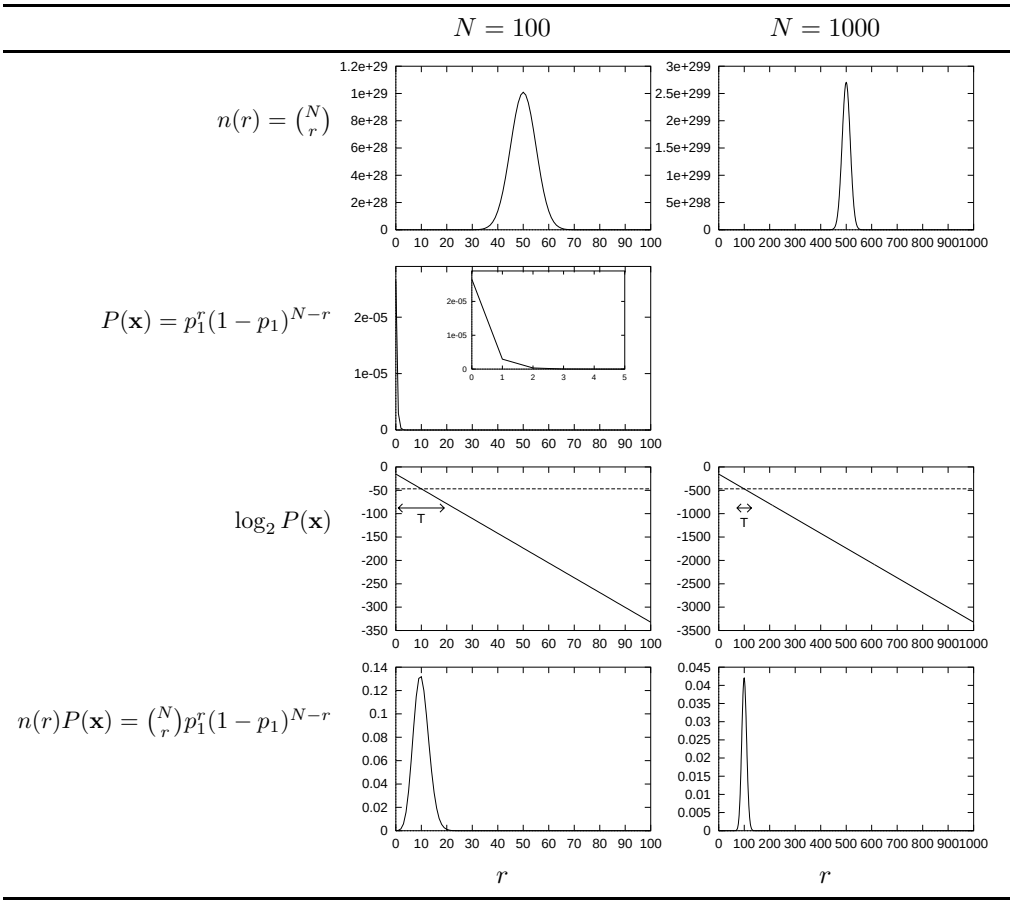


Figure 4.11. Anatomy of the typical set T. For $p_1 = 0.1$ and $N = 100$ and $N = 1000$, these graphs show $n(r)$, the number of strings containing r 1s; the probability $P(\mathbf{x})$ of a single string that contains r 1s; the same probability on a log scale; and the total probability $n(r)P(\mathbf{x})$ of all strings that contain r 1s. The number r is on the horizontal axis. The plot of $\log_2 P(\mathbf{x})$ also shows by a dotted line the mean value of $\log_2 P(\mathbf{x}) = -NH_2(p_1)$ which equals -46.9 when $N = 100$ and -469 when $N = 1000$. The typical set includes only the strings that have $\log_2 P(\mathbf{x})$ close to this value. The range marked T shows the set $T_{N\beta}$ (as defined in section 4.4) for $N = 100$ and $\beta = 0.29$ (left) and $N = 1000$, $\beta = 0.09$ (right).

If $N = 1000$ then

$$r \sim 100 \pm 10. \tag{4.26}$$

Notice that as N gets bigger, the probability distribution of r becomes more concentrated, in the sense that while the range of possible values of r grows as N, the standard deviation of r only grows as $\sqrt{N}$. That r is most likely to fall in a small range of values implies that the outcome $\mathbf{x}$ is also most likely to fall in a corresponding small subset of outcomes that we will call the *typical set*.

Definition of the typical set

Let us define typicality for an arbitrary ensemble X with alphabet $\mathcal{A}_X$. Our definition of a typical string will involve the string's probability. A long string of N symbols will usually contain about $p_1 N$ occurrences of the first symbol, $p_2 N$ occurrences of the second, etc. Hence the probability of this string is roughly

$$P(\mathbf{x})_{\text{typ}} = P(x_1)P(x_2)P(x_3)\ldots P(x_N) \simeq p_1^{(p_1 N)} p_2^{(p_2 N)} \ldots p_I^{(p_I N)} \tag{4.27}$$

so that the information content of a typical string is

$$\log_2 \frac{1}{P(\mathbf{x})} \simeq N \sum_i p_i \log_2 \frac{1}{p_i} \simeq NH. \tag{4.28}$$

So the random variable $\log_2 {}^{1}/_{P(\mathbf{x})}$, which is the information content of $\mathbf{x}$, is very likely to be close in value to NH. We build our definition of typicality on this observation.

We define the typical elements of $\mathcal{A}_X^N$ to be those elements that have probability close to 2^{-NH}. (Note that the typical set, unlike the smallest sufficient subset, does *not* include the most probable elements of $\mathcal{A}_X^N$, but we will show that these most probable elements contribute negligible probability.)

We introduce a parameter β that defines how close the probability has to be to 2^{-NH} for an element to be 'typical'. We call the set of typical elements the typical set, $T_{N\beta}$:

$$T_{N\beta} \equiv \left\{ \mathbf{x} \in \mathcal{A}_X^N : \left| \frac{1}{N} \log_2 \frac{1}{P(\mathbf{x})} - H \right| < \beta \right\}. \tag{4.29}$$

We will show that whatever value of β we choose, the typical set contains almost all the probability as N increases.

This important result is sometimes called the *'asymptotic equipartition' principle*.

'Asymptotic equipartition' principle. For an ensemble of N independent identically distributed (i.i.d.) random variables $X^N \equiv (X_1, X_2, \ldots, X_N)$, with N sufficiently large, the outcome $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ is almost certain to belong to a subset of $\mathcal{A}_X^N$ having only $2^{NH(X)}$ members, each having probability 'close to' $2^{-NH(X)}$.

Notice that if $H(X) < H_0(X)$ then $2^{NH(X)}$ is a *tiny* fraction of the number of possible outcomes $|\mathcal{A}_X^N| = |\mathcal{A}_X|^N = 2^{NH_0(X)}$.

> The term equipartition is chosen to describe the idea that the members of the typical set have *roughly equal* probability. [This should not be taken too literally, hence my use of quotes around 'asymptotic equipartition'; see page 83.]
>
> A second meaning for equipartition, in thermal physics, is the idea that each degree of freedom of a classical system has equal average energy, $\frac{1}{2}kT$. This second meaning is not intended here.

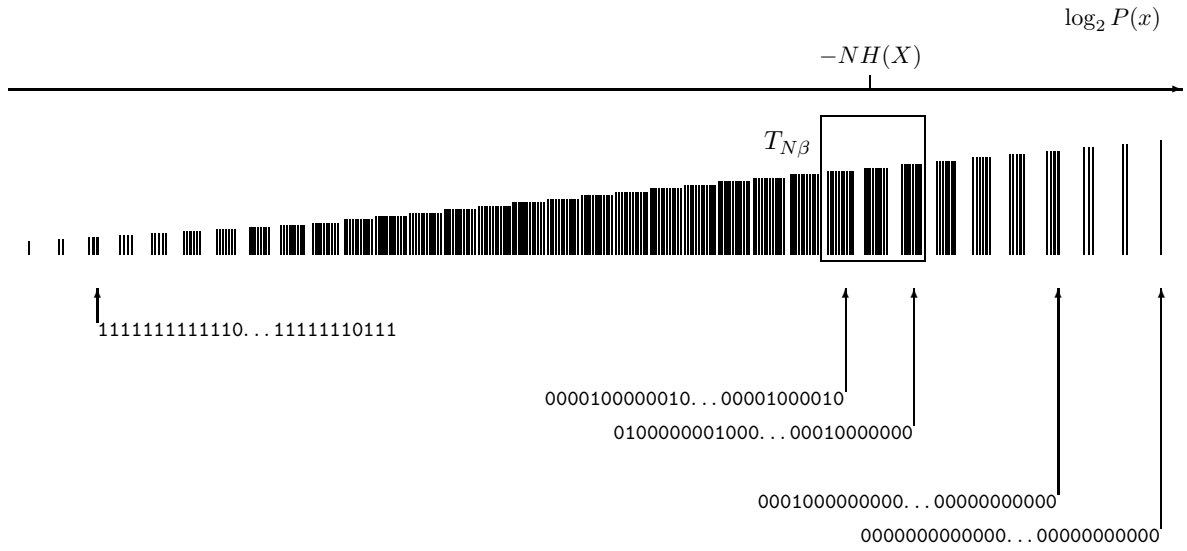


Figure 4.12. Schematic diagram showing all strings in the ensemble X^N ranked by their probability, and the typical set $T_{N\beta}$.

The 'asymptotic equipartition' principle is equivalent to:

Shannon's source coding theorem (verbal statement). N i.i.d. random variables each with entropy $H(X)$ can be compressed into more than $NH(X)$ bits with negligible risk of information loss, as $N \to \infty$; conversely if they are compressed into fewer than $NH(X)$ bits it is virtually certain that information will be lost.

These two theorems are equivalent because we can define a compression algorithm that gives a distinct name of length $NH(X)$ bits to each $\mathbf{x}$ in the typical set.

▶ 4.5 Proofs

This section may be skipped if found tough going.

The law of large numbers

Our proof of the source coding theorem uses the law of large numbers.

Mean and variance of a real random variable are $\mathcal{E}[u] = \bar{u} = \sum_u P(u)u$ and $\mathrm{var}(u) = \sigma_u^2 = \mathcal{E}[(u-\bar{u})^2] = \sum_u P(u)(u-\bar{u})^2$.

> Technical note: strictly I am assuming here that u is a function $u(x)$ of a sample x from a finite discrete ensemble X. Then the summations $\sum_u P(u)f(u)$ should be written $\sum_x P(x)f(u(x))$. This means that $P(u)$ is a finite sum of delta functions. This restriction guarantees that the mean and variance of u do exist, which is not necessarily the case for general $P(u)$.

Chebyshev's inequality 1. Let t be a non-negative real random variable, and let α be a positive real number. Then

$$P(t \geq \alpha) \leq \frac{\bar{t}}{\alpha}. \tag{4.30}$$

> Proof: $P(t \geq \alpha) = \sum_{t \geq \alpha} P(t)$. We multiply each term by $t/\alpha \geq 1$ and obtain: $P(t \geq \alpha) \leq \sum_{t \geq \alpha} P(t)t/\alpha$. We add the (non-negative) missing terms and obtain: $P(t \geq \alpha) \leq \sum_t P(t)t/\alpha = \bar{t}/\alpha$. □

Chebyshev's inequality 2. Let x be a random variable, and let α be a positive real number. Then

$$P\left((x-\bar{x})^2 \geq \alpha\right) \leq \sigma_x^2/\alpha. \tag{4.31}$$

Proof: Take $t = (x-\bar{x})^2$ and apply the previous proposition. □

Weak law of large numbers. Take x to be the average of N independent random variables $h_1, \ldots, h_N$, having common mean $\bar{h}$ and common variance σ_h^2: $x = \frac{1}{N}\sum_{n=1}^{N} h_n$. Then

$$P((x-\bar{h})^2 \geq \alpha) \leq \sigma_h^2/\alpha N. \tag{4.32}$$

Proof: obtained by showing that $\bar{x} = \bar{h}$ and that $\sigma_x^2 = \sigma_h^2/N$. □

We are interested in x being very close to the mean (α very small). No matter how large σ_h^2 is, and no matter how small the required α is, and no matter how small the desired probability that $(x-\bar{h})^2 \geq \alpha$, we can always achieve it by taking N large enough.

Proof of theorem 4.1 (p.78)

We apply the law of large numbers to the random variable $\frac{1}{N}\log_2 \frac{1}{P(\mathbf{x})}$ defined for $\mathbf{x}$ drawn from the ensemble X^N. This random variable can be written as the average of N information contents $h_n = \log_2(1/P(x_n))$, each of which is a random variable with mean $H = H(X)$ and variance $\sigma^2 \equiv \mathrm{var}[\log_2(1/P(x_n))]$. (Each term h_n is the Shannon information content of the nth outcome.)

We again define the typical set with parameters N and β thus:

$$T_{N\beta} = \left\{ \mathbf{x} \in \mathcal{A}_X^N : \left[\frac{1}{N}\log_2 \frac{1}{P(\mathbf{x})} - H \right]^2 < \beta^2 \right\}. \tag{4.33}$$

For all $\mathbf{x} \in T_{N\beta}$, the probability of $\mathbf{x}$ satisfies

$$2^{-N(H+\beta)} < P(\mathbf{x}) < 2^{-N(H-\beta)}. \tag{4.34}$$

And by the law of large numbers,

$$P(\mathbf{x} \in T_{N\beta}) \geq 1 - \frac{\sigma^2}{\beta^2 N}. \tag{4.35}$$

We have thus proved the 'asymptotic equipartition' principle. As N increases, the probability that $\mathbf{x}$ falls in $T_{N\beta}$ approaches 1, for any β. How does this result relate to source coding?

We must relate $T_{N\beta}$ to $H_\delta(X^N)$. We will show that for any given δ there is a sufficiently big N such that $H_\delta(X^N) \simeq NH$.

Part 1: $\frac{1}{N}H_\delta(X^N) < H + \epsilon$.

The set $T_{N\beta}$ is not the best subset for compression. So the size of $T_{N\beta}$ gives an upper bound on H_δ. We show how *small* $H_\delta(X^N)$ must be by calculating how big $T_{N\beta}$ could possibly be. We are free to set β to any convenient value. The smallest possible probability that a member of $T_{N\beta}$ can have is $2^{-N(H+\beta)}$, and the total probability that $T_{N\beta}$ contains can't be any bigger than 1. So

$$|T_{N\beta}|\, 2^{-N(H+\beta)} < 1, \tag{4.36}$$

that is, the size of the typical set is bounded by

$$|T_{N\beta}| < 2^{N(H+\beta)}. \tag{4.37}$$

If we set $\beta = \epsilon$ and N_0 such that $\frac{\sigma^2}{\epsilon^2 N} \leq \delta$, then $P(T_{N\beta}) \geq 1-\delta$, and the set $T_{N\beta}$ becomes a witness to the fact that $H_\delta(X^N) \leq \log_2 |T_{N\beta}| < N(H+\epsilon)$.

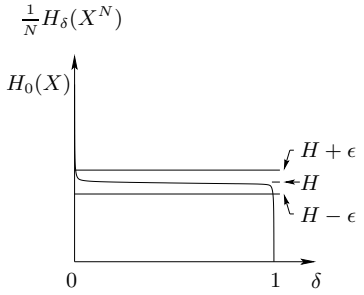
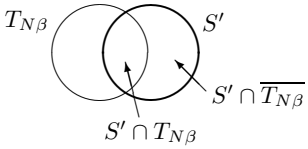


Figure 4.13. Schematic illustration of the two parts of the theorem. Given any δ and ϵ, we show that for large enough N, $\frac{1}{N}H_\delta(X^N)$ lies (1) below the line $H+\epsilon$ and (2) above the line $H-\epsilon$.

Part 2: $\frac{1}{N}H_\delta(X^N) > H - \epsilon$.

Imagine that someone claims this second part is not so – that, for any N, the smallest δ-sufficient subset S_δ is smaller than the above inequality would allow. We can make use of our typical set to show that they must be mistaken. Remember that we are free to set β to any value we choose. We will set $\beta = \epsilon/2$, so that our task is to prove that a subset S' having $|S'| \leq 2^{N(H-2\beta)}$ and achieving $P(\mathbf{x} \in S') \geq 1 - \delta$ cannot exist (for N greater than an N_0 that we will specify).

So, let us consider the probability of falling in this rival smaller subset S'. The probability of the subset S' is



$$P(\mathbf{x} \in S') \;=\; P(\mathbf{x} \in S' \cap T_{N\beta}) + P(\mathbf{x} \in S' \cap \overline{T_{N\beta}}), \tag{4.38}$$

where $\overline{T_{N\beta}}$ denotes the complement $\{\mathbf{x} \notin T_{N\beta}\}$. The maximum value of the first term is found if $S' \cap T_{N\beta}$ contains $2^{N(H-2\beta)}$ outcomes all with the maximum probability, $2^{-N(H-\beta)}$. The maximum value the second term can have is $P(\mathbf{x} \notin T_{N\beta})$. So:

$$P(\mathbf{x} \in S') \;\leq\; 2^{N(H-2\beta)}\, 2^{-N(H-\beta)} + \frac{\sigma^2}{\beta^2 N} = 2^{-N\beta} + \frac{\sigma^2}{\beta^2 N}. \tag{4.39}$$

We can now set $\beta = \epsilon/2$ and N_0 such that $P(\mathbf{x} \in S') < 1 - \delta$, which shows that S' cannot satisfy the definition of a sufficient subset S_δ. Thus *any* subset S' with size $|S'| \leq 2^{N(H-\epsilon)}$ has probability less than $1 - \delta$, so by the definition of H_δ, $H_\delta(X^N) > N(H - \epsilon)$.

Thus for large enough N, the function $\frac{1}{N}H_\delta(X^N)$ is essentially a constant function of δ, for $0 < \delta < 1$, as illustrated in figures 4.9 and 4.13. □

▶ 4.6 Comments

The source coding theorem (p.78) has two parts, $\frac{1}{N}H_\delta(X^N) < H + \epsilon$, and $\frac{1}{N}H_\delta(X^N) > H - \epsilon$. Both results are interesting.

The first part tells us that even if the probability of error δ is extremely small, the number of bits per symbol $\frac{1}{N}H_\delta(X^N)$ needed to specify a long N-symbol string $\mathbf{x}$ with vanishingly small error probability does not have to exceed $H + \epsilon$ bits. We need to have only a tiny tolerance for error, and the number of bits required drops significantly from $H_0(X)$ to $(H + \epsilon)$.

What happens if we are yet more tolerant to compression errors? Part 2 tells us that even if δ is very close to 1, so that errors are made most of the time, the average number of bits per symbol needed to specify $\mathbf{x}$ must still be at least $H - \epsilon$ bits. These two extremes tell us that regardless of our specific allowance for error, the number of bits per symbol needed to specify $\mathbf{x}$ is H bits; no more and no less.

Caveat regarding 'asymptotic equipartition'

I put the words 'asymptotic equipartition' in quotes because it is important not to think that the elements of the typical set $T_{N\beta}$ really do have roughly the same probability as each other. They are similar in probability only in the sense that their values of $\log_2 \frac{1}{P(\mathbf{x})}$ are within $2N\beta$ of each other. Now, as β is decreased, how does N have to increase, if we are to keep our bound on the mass of the typical set, $P(\mathbf{x} \in T_{N\beta}) \geq 1 - \frac{\sigma^2}{\beta^2 N}$, constant? N must grow as $1/\beta^2$, so, if we write β in terms of N as $\alpha/\sqrt{N}$, for some constant α, then

the most probable string in the typical set will be of order $2^{\alpha\sqrt{N}}$ times greater than the least probable string in the typical set. As β decreases, N increases, and this ratio $2^{\alpha\sqrt{N}}$ grows exponentially. Thus we have 'equipartition' only in a weak sense!

Why did we introduce the typical set?

The best choice of subset for block compression is (by definition) S_δ, not a typical set. So why did we bother introducing the typical set? The answer is, *we can count the typical set.* We know that all its elements have 'almost identical' probability (2^{-NH}), and we know the whole set has probability almost 1, so the typical set must have roughly 2^{NH} elements. Without the help of the typical set (which is very similar to S_δ) it would have been hard to count how many elements there are in S_δ.

4.7 Exercises

Weighing problems

▷ Exercise 4.9.[1] While some people, when they first encounter the weighing problem with 12 balls and the three-outcome balance (exercise 4.1 (p.66)), think that weighing six balls against six balls is a good first weighing, others say 'no, weighing six against six conveys *no* information at all'. Explain to the second group why they are both right and wrong. Compute the information gained about *which is the odd ball*, and the information gained about *which is the odd ball and whether it is heavy or light.*

▷ Exercise 4.10.[2] Solve the weighing problem for the case where there are 39 balls of which one is known to be odd.

▷ Exercise 4.11.[2] You are given 16 balls, all of which are equal in weight except for one that is either heavier or lighter. You are also given a bizarre two-pan balance that can report only two outcomes: 'the two sides balance' or 'the two sides do not balance'. Design a strategy to determine which is the odd ball in as few uses of the balance as possible.

▷ Exercise 4.12.[2] You have a two-pan balance; your job is to weigh out bags of flour with integer weights 1 to 40 pounds inclusive. How many weights do you need? [You are allowed to put weights on either pan. You're only allowed to put one flour bag on the balance at a time.]

Exercise 4.13.[4, p.86] (a) Is it possible to solve exercise 4.1 (p.66) (the weighing problem with 12 balls and the three-outcome balance) using a sequence of three *fixed* weighings, such that the balls chosen for the second weighing do not depend on the outcome of the first, and the third weighing does not depend on the first or second?

(b) Find a solution to the general N-ball weighing problem in which exactly one of N balls is odd. Show that in W weighings, an odd ball can be identified from among $N = (3^W - 3)/2$ balls.

Exercise 4.14.[3] You are given 12 balls and the three-outcome balance of exercise 4.1; this time, *two* of the balls are odd; each odd ball may be heavy or light, and we don't know which. We want to identify the odd balls and in which direction they are odd.

(a) *Estimate* how many weighings are required by the optimal strategy. And what if there are three odd balls?

(b) How do your answers change if it is known that all the regular balls weigh 100 g, that light balls weigh 99 g, and heavy ones weigh 110 g?

Source coding with a lossy compressor, with loss δ

▷ Exercise 4.15.[2, p.87] Let $\mathcal{P}_X = \{0.2, 0.8\}$. Sketch $\frac{1}{N}H_\delta(X^N)$ as a function of δ for $N = 1, 2$ and 1000.

▷ Exercise 4.16.[2] Let $\mathcal{P}_Y = \{0.5, 0.5\}$. Sketch $\frac{1}{N}H_\delta(Y^N)$ as a function of δ for $N = 1, 2, 3$ and 100.

▷ Exercise 4.17.[2, p.87] (For physics students.) Discuss the relationship between the proof of the 'asymptotic equipartition' principle and the equivalence (for large systems) of the Boltzmann entropy and the Gibbs entropy.

Distributions that don't obey the law of large numbers

The law of large numbers, which we used in this chapter, shows that the mean of a set of N i.i.d. random variables has a probability distribution that becomes narrower, with width $\propto 1/\sqrt{N}$, as N increases. However, we have proved this property only for discrete random variables, that is, for real numbers taking on a *finite* set of possible values. While many random variables with continuous probability distributions also satisfy the law of large numbers, there are important distributions that do not. Some continuous distributions do not have a mean or variance.

▷ Exercise 4.18.[3, p.88] Sketch the Cauchy distribution

$$P(x) = \frac{1}{Z}\frac{1}{x^2 + 1}, \quad x \in (-\infty, \infty). \tag{4.40}$$

What is its normalizing constant Z? Can you evaluate its mean or variance?

Consider the sum $z = x_1 + x_2$, where x_1 and x_2 are independent random variables from a Cauchy distribution. What is $P(z)$? What is the probability distribution of the mean of x_1 and x_2, $\bar{x} = (x_1 + x_2)/2$? What is the probability distribution of the mean of N samples from this Cauchy distribution?

Other asymptotic properties

Exercise 4.19.[3] Chernoff bound. We derived the weak law of large numbers from Chebyshev's inequality (4.30) by letting the random variable t in the inequality $P(t \geq \alpha) \leq \bar{t}/\alpha$ be a function, $t = (x - \bar{x})^2$, of the random variable x we were interested in.

Other useful inequalities can be obtained by using other functions. The Chernoff bound, which is useful for bounding the tails of a distribution, is obtained by letting $t = \exp(sx)$.

Show that

$$P(x \geq a) \leq e^{-sa} g(s), \quad \text{for any } s > 0 \tag{4.41}$$

and

$$P(x \leq a) \leq e^{-sa} g(s), \quad \text{for any } s < 0 \tag{4.42}$$

where $g(s)$ is the moment-generating function of x,

$$g(s) = \sum_x P(x)\, e^{sx}. \tag{4.43}$$

Curious functions related to $p \log 1/p$

Exercise 4.20.[4, p.89] This exercise has no purpose at all; it's included for the enjoyment of those who like mathematical curiosities.

Sketch the function

$$f(x) = x^{x^{x^{x^{\cdot^{\cdot}}}}} \tag{4.44}$$

for $x \geq 0$. Hint: Work out the inverse function to f – that is, the function $g(y)$ such that if $x = g(y)$ then $y = f(x)$ – it's closely related to $p \log 1/p$.

4.8 Solutions

Solution to exercise 4.2 (p.68). Let $P(x, y) = P(x)P(y)$. Then

$$\begin{aligned} H(X,Y) &= \sum_{xy} P(x)P(y) \log \frac{1}{P(x)P(y)} && (4.45)\\ &= \sum_{xy} P(x)P(y) \log \frac{1}{P(x)} + \sum_{xy} P(x)P(y) \log \frac{1}{P(y)} && (4.46)\\ &= \sum_{x} P(x) \log \frac{1}{P(x)} + \sum_{y} P(y) \log \frac{1}{P(y)} && (4.47)\\ &= H(X) + H(Y). && (4.48) \end{aligned}$$

Solution to exercise 4.4 (p.73). An ASCII file can be reduced in size by a factor of 7/8. This reduction could be achieved by a block code that maps 8-byte blocks into 7-byte blocks by copying the 56 information-carrying bits into 7 bytes, and ignoring the last bit of every character.

Solution to exercise 4.5 (p.74). The pigeon-hole principle states: you can't put 16 pigeons into 15 holes without using one of the holes twice.

Similarly, you can't give $\mathcal{A}_X$ outcomes unique binary names of some length l shorter than $\log_2 |\mathcal{A}_X|$ bits, because there are only 2^l such binary names, and $l < \log_2 |\mathcal{A}_X|$ implies $2^l < |\mathcal{A}_X|$, so at least two different inputs to the compressor would compress to the same output file.

Solution to exercise 4.8 (p.76). Between the cusps, all the changes in probability are equal, and the number of elements in T changes by one at each step. So H_δ varies logarithmically with $(-\delta)$.

Solution to exercise 4.13 (p.84). This solution was found by Dyson and Lyness in 1946 and presented in the following elegant form by John Conway in 1999. Be warned: the symbols A, B, and C are used to name the balls, to name the pans of the balance, to name the outcomes, and to name the possible states of the odd ball!

(a) Label the 12 balls by the sequences

```
AAB  ABA  ABB  ABC  BBC  BCA  BCB  BCC  CAA  CAB  CAC  CCA
```

and in the