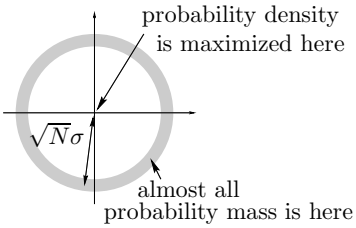

Figure 6.8. Schematic representation of the typical set of an N-dimensional Gaussian distribution.

Assuming that N is large, show that nearly all the probability of a Gaussian is contained in a thin shell of radius $\sqrt{N}\sigma$. Find the thickness of the shell.

Evaluate the probability density (6.13) at a point in that thin shell and at the origin $\mathbf{x} = 0$ and compare. Use the case $N = 1000$ as an example.

Notice that nearly all the probability mass is located in a different part of the space from the region of highest probability density.

Exercise 6.15.[2] Explain what is meant by an *optimal binary symbol code*.

Find an optimal binary symbol code for the ensemble:

$$\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}\},$$

$$\mathcal{P} = \left\{ \frac{1}{100}, \frac{2}{100}, \frac{4}{100}, \frac{5}{100}, \frac{6}{100}, \frac{8}{100}, \frac{9}{100}, \frac{10}{100}, \frac{25}{100}, \frac{30}{100} \right\},$$

and compute the expected length of the code.

Exercise 6.16.[2] A string $\mathbf{y} = x_1 x_2$ consists of *two* independent samples from an ensemble

$$X : \mathcal{A}_X = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}; \mathcal{P}_X = \left\{ \frac{1}{10}, \frac{3}{10}, \frac{6}{10} \right\}.$$

What is the entropy of $\mathbf{y}$? Construct an optimal binary symbol code for the string $\mathbf{y}$, and find its expected length.

Exercise 6.17.[2] Strings of N independent samples from an ensemble with $\mathcal{P} = \{0.1, 0.9\}$ are compressed using an arithmetic code that is matched to that ensemble. Estimate the mean and standard deviation of the compressed strings' lengths for the case $N = 1000$. $[H_2(0.1) \simeq 0.47]$

Exercise 6.18.[3] Source coding with variable-length symbols.

> In the chapters on source coding, we assumed that we were encoding into a binary alphabet $\{\mathtt{0}, \mathtt{1}\}$ in which both symbols should be used with equal frequency. In this question we explore how the encoding alphabet should be used if the symbols take different times to transmit.

A poverty-stricken student communicates for free with a friend using a telephone by selecting an integer $n \in \{1, 2, 3 \ldots\}$, making the friend's phone ring n times, then hanging up in the middle of the nth ring. This process is repeated so that a string of symbols $n_1 n_2 n_3 \ldots$ is received. What is the optimal way to communicate? If large integers n are selected then the message takes longer to communicate. If only small integers n are used then the information content per symbol is small. We aim to maximize the rate of information transfer, per unit time.

Assume that the time taken to transmit a number of rings n and to redial is l_n seconds. Consider a probability distribution over n, $\{p_n\}$. Defining the average duration *per symbol* to be

$$L(\mathbf{p}) = \sum_n p_n l_n \tag{6.15}$$

and the entropy *per symbol* to be

$$H(\mathbf{p}) = \sum_n p_n \log_2 \frac{1}{p_n}, \tag{6.16}$$

show that for the average information rate *per second* to be maximized, the symbols must be used with probabilities of the form

$$p_n = \frac{1}{Z} 2^{-\beta l_n} \tag{6.17}$$

where $Z = \sum_n 2^{-\beta l_n}$ and β satisfies the implicit equation

$$\beta = \frac{H(\mathbf{p})}{L(\mathbf{p})}, \tag{6.18}$$

that is, β is the rate of communication. Show that these two equations (6.17, 6.18) imply that β must be set such that

$$\log Z = 0. \tag{6.19}$$

Assuming that the channel has the property

$$l_n = n \text{ seconds}, \tag{6.20}$$

find the optimal distribution $\mathbf{p}$ and show that the maximal information rate is 1 bit per second.

How does this compare with the information rate per second achieved if $\mathbf{p}$ is set to $(1/2, 1/2, 0, 0, 0, 0, \ldots)$ — that is, only the symbols $n = 1$ and $n = 2$ are selected, and they have equal probability?

Discuss the relationship between the results (6.17, 6.19) derived above, and the Kraft inequality from source coding theory.

How might a random binary source be efficiently encoded into a sequence of symbols $n_1 n_2 n_3 \ldots$ for transmission over the channel defined in equation (6.20)?

▷ Exercise 6.19.[1] How many bits does it take to shuffle a pack of cards?

▷ Exercise 6.20.[2] In the card game Bridge, the four players receive 13 cards each from the deck of 52 and start each game by looking at their own hand and bidding. The legal bids are, in ascending order $1\clubsuit, 1\diamondsuit, 1\heartsuit, 1\spadesuit$, $1NT$, $2\clubsuit$, $2\diamondsuit$, ... $7\heartsuit, 7\spadesuit, 7NT$, and successive bids must follow this order; a bid of, say, $2\heartsuit$ may only be followed by higher bids such as $2\spadesuit$ or $3\clubsuit$ or $7NT$. (Let us neglect the 'double' bid.)

The players have several aims when bidding. One of the aims is for two partners to communicate to each other as much as possible about what cards are in their hands.

Let us concentrate on this task.

(a) After the cards have been dealt, how many bits are needed for North to convey to South what her hand is?

(b) Assuming that E and W do not bid at all, what is the maximum total information that N and S can convey to each other while bidding? Assume that N starts the bidding, and that once either N or S stops bidding, the bidding stops.

▷ Exercise 6.21.[2] My old 'arabic' microwave oven had 11 buttons for entering cooking times, and my new 'roman' microwave has just five. The buttons of the roman microwave are labelled '10 minutes', '1 minute', '10 seconds', '1 second', and 'Start'; I'll abbreviate these five strings to the symbols `M`, `C`, `X`, `I`, □. To enter one minute and twenty-three seconds (1:23), the arabic sequence is

$$\texttt{123}\square, \tag{6.21}$$

and the roman sequence is

$$\texttt{CXXIII}\square. \tag{6.22}$$

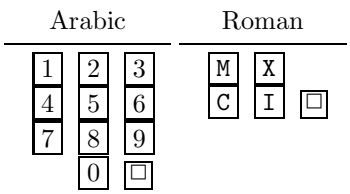

Figure 6.9. Alternative keypads for microwave ovens.

Each of these keypads defines a code mapping the 3599 cooking times from 0:01 to 59:59 into a string of symbols.

(a) Which times can be produced with two or three symbols? (For example, 0:20 can be produced by three symbols in either code: `XX`□ and `20`□.)

(b) Are the two codes complete? Give a detailed answer.

(c) For each code, name a cooking time that it can produce in four symbols that the other code cannot.

(d) Discuss the implicit probability distributions over times to which each of these codes is best matched.

(e) Concoct a plausible probability distribution over times that a real user might use, and evaluate roughly the expected number of symbols, and maximum number of symbols, that each code requires. Discuss the ways in which each code is inefficient or efficient.

(f) Invent a more efficient cooking-time-encoding system for a microwave oven.

Exercise 6.22.[2, p.132] Is the standard binary representation for positive integers (e.g. $c_{\mathrm{b}}(5) = \texttt{101}$) a uniquely decodeable code?

Design a binary code for the positive integers, i.e., a mapping from $n \in \{1, 2, 3, \ldots\}$ to $c(n) \in \{\texttt{0},\texttt{1}\}^+$, that is uniquely decodeable. Try to design codes that are prefix codes and that satisfy the Kraft equality $\sum_n 2^{-l_n} = 1$.

> Motivations: any data file terminated by a special end of file character can be mapped onto an integer, so a prefix code for integers can be used as a self-delimiting encoding of files too. Large files correspond to large integers. Also, one of the building blocks of a 'universal' coding scheme – that is, a coding scheme that will work OK for a large variety of sources – is the ability to encode integers. Finally, in microwave ovens, cooking times are positive integers!

Discuss criteria by which one might compare alternative codes for integers (or, equivalently, alternative self-delimiting codes for files).

▶ 6.9 Solutions

Solution to exercise 6.1 (p.115). The worst-case situation is when the interval to be represented lies just inside a binary interval. In this case, we may choose either of two binary intervals as shown in figure 6.10. These binary intervals

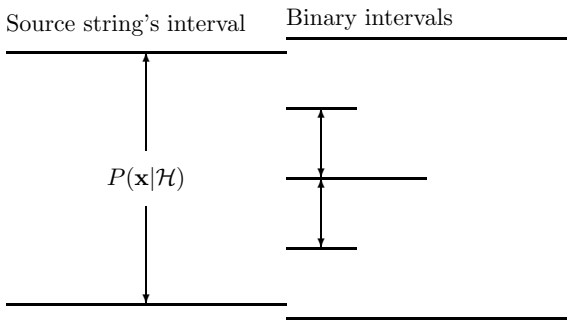

Figure 6.10. Termination of arithmetic coding in the worst case, where there is a two bit overhead. Either of the two binary intervals marked on the right-hand side may be chosen. These binary intervals are no smaller than $P(\mathbf{x}|\mathcal{H})/4$.

are no smaller than $P(\mathbf{x}|\mathcal{H})/4$, so the binary encoding has a length no greater than $\log_2 1/P(\mathbf{x}|\mathcal{H}) + \log_2 4$, which is two bits more than the ideal message length.

Solution to exercise 6.3 (p.118). The standard method uses 32 random bits per generated symbol and so requires 32 000 bits to generate one thousand samples.

Arithmetic coding uses on average about $H_2(0.01) = 0.081$ bits per generated symbol, and so requires about 83 bits to generate one thousand samples (assuming an overhead of roughly two bits associated with termination).

Fluctuations in the number of 1s would produce variations around this mean with standard deviation 21.

Solution to exercise 6.5 (p.120). The encoding is 010100110010110001100, which comes from the parsing

$$0, 00, 000, 0000, 001, 00000, 000000 \tag{6.23}$$

which is encoded thus:

$$(, 0), (1, 0), (10, 0), (11, 0), (010, 1), (100, 0), (110, 0). \tag{6.24}$$

Solution to exercise 6.6 (p.120). The decoding is
0100001000100010101000001.

Solution to exercise 6.8 (p.123). This problem is equivalent to exercise 6.7 (p.123).

The selection of K objects from N objects requires $\lceil \log_2 \binom{N}{K} \rceil$ bits $\simeq NH_2(K/N)$ bits. This selection could be made using arithmetic coding. The selection corresponds to a binary string of length N in which the 1 bits represent which objects are selected. Initially the probability of a 1 is K/N and the probability of a 0 is $(N-K)/N$. Thereafter, given that the emitted string thus far, of length n, contains k 1s, the probability of a 1 is $(K-k)/(N-n)$ and the probability of a 0 is $1 - (K-k)/(N-n)$.

Solution to exercise 6.12 (p.124). This modified Lempel–Ziv code is still not 'complete', because, for example, after five prefixes have been collected, the pointer could be any of the strings 000, 001, 010, 011, 100, but it cannot be 101, 110 or 111. Thus there are some binary strings that cannot be produced as encodings.

Solution to exercise 6.13 (p.124). Sources with low entropy that are not well compressed by Lempel–Ziv include:

(a) Sources with some symbols that have long range correlations and intervening random junk. An ideal model should capture what's correlated and compress it. Lempel–Ziv can compress the correlated features only by memorizing all cases of the intervening junk. As a simple example, consider a telephone book in which every line contains an (old number, new number) pair:

```
285-3820:572-5892□
258-8302:593-2010□
```

The number of characters per line is 18, drawn from the 13-character alphabet $\{0, 1, \ldots, 9, -, :, \square\}$. The characters '-', ':' and '□' occur in a predictable sequence, so the true information content per line, assuming all the phone numbers are seven digits long, and assuming that they are random sequences, is about 14 bans. (A ban is the information content of a random integer between 0 and 9.) A finite state language model could easily capture the regularities in these data. A Lempel–Ziv algorithm will take a long time before it compresses such a file down to 14 bans per line, however, because in order for it to 'learn' that the string `:`*ddd* is always followed by `-`, for any three digits *ddd*, it will have to *see* all those strings. So near-optimal compression will only be achieved after thousands of lines of the file have been read.

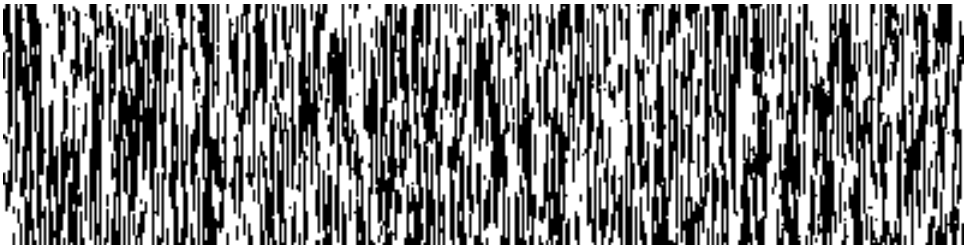

Figure 6.11. A source with low entropy that is not well compressed by Lempel–Ziv. The bit sequence is read from left to right. Each line differs from the line above in $f = 5\%$ of its bits. The image width is 400 pixels.

(b) Sources with long range correlations, for example two-dimensional images that are represented by a sequence of pixels, row by row, so that vertically adjacent pixels are a distance w apart in the source stream, where w is the image width. Consider, for example, a fax transmission in which each line is very similar to the previous line (figure 6.11). The true entropy is only $H_2(f)$ per pixel, where f is the probability that a pixel differs from its parent. Lempel–Ziv algorithms will only compress down to the entropy once *all* strings of length $2^w = 2^{400}$ have occurred and their successors have been memorized. There are only about 2^{300} particles in the universe, so we can confidently say that Lempel–Ziv codes will *never* capture the redundancy of such an image.

Another highly redundant texture is shown in figure 6.12. The image was made by dropping horizontal and vertical pins randomly on the plane. It contains both long-range vertical correlations and long-range horizontal correlations. There is no practical way that Lempel–Ziv, fed with a pixel-by-pixel scan of this image, could capture both these correlations.

Biological computational systems can readily identify the redundancy in these images and in images that are much more complex; thus we might anticipate that the best data compression algorithms will result from the development of artificial intelligence methods.

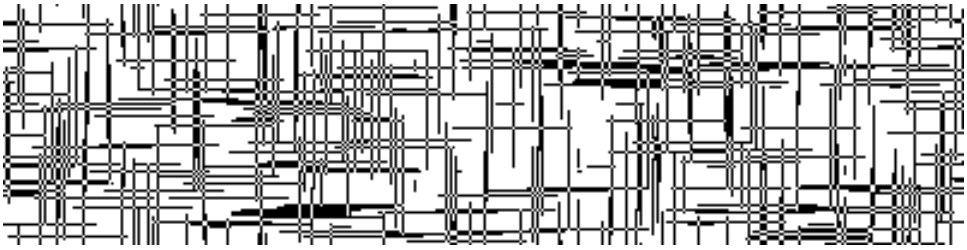

Figure 6.12. A texture consisting of horizontal and vertical pins dropped at random on the plane.

(c) Sources with intricate redundancy, such as files generated by computers. For example, a LaTeX file followed by its encoding into a PostScript file. The information content of this pair of files is roughly equal to the information content of the LaTeX file alone.

(d) A picture of the Mandelbrot set. The picture has an information content equal to the number of bits required to specify the range of the complex plane studied, the pixel sizes, and the colouring rule used.

(e) A picture of a ground state of a frustrated antiferromagnetic Ising model (figure 6.13), which we will discuss in Chapter 31. Like figure 6.12, this binary image has interesting correlations in two directions.

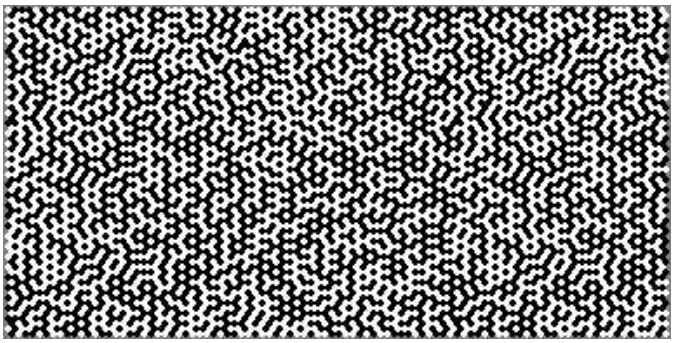

Figure 6.13. Frustrated triangular Ising model in one of its ground states.

(f) Cellular automata – figure 6.14 shows the state history of 100 steps of a cellular automaton with 400 cells. The update rule, in which each cell's new state depends on the state of five preceding cells, was selected at random. The information content is equal to the information in the boundary (400 bits), and the propagation rule, which here can be described in 32 bits. An optimal compressor will thus give a compressed file length which is essentially constant, independent of the vertical height of the image. Lempel–Ziv would only give this zero-cost compression once the cellular automaton has entered a periodic limit cycle, which could easily take about 2^{100} iterations.

In contrast, the JBIG compression method, which models the probability of a pixel given its local context and uses arithmetic coding, would do a good job on these images.

Solution to exercise 6.14 (p.124). For a one-dimensional Gaussian, the variance of x, $\mathcal{E}[x^2]$, is σ^2. So the mean value of r^2 in N dimensions, since the components of $\mathbf{x}$ are independent random variables, is

$$\mathcal{E}[r^2] = N\sigma^2. \tag{6.25}$$