Extra Solutions to Exercises

This is the solutions manual for *Information Theory, Inference, and Learning Algorithms*. It is supplied on request to instructors using this book in their teaching; please email `solutions@cambridge.org`. For the benefit of instructors, please do not circulate this document to students.

Please send corrections or additions to these solutions to David MacKay, `mackay@mrao.cam.ac.uk`.

Reminder about internet resources

The website

`http://www.inference.phy.cam.ac.uk/mackay/itila`

contains several resources for this book.

Extra Solutions for Chapter 1

Solution to exercise 1.4 (p.12). The matrix $\mathbf{HG}^{\mathsf{T}} \bmod 2$ is equal to the all-zero 3×4 matrix, so for any codeword $\mathbf{t} = \mathbf{G}^{\mathsf{T}}\mathbf{s}$, $\mathbf{Ht} = \mathbf{HG}^{\mathsf{T}}\mathbf{s} = (0,0,0)^{\mathsf{T}}$.

Solution to exercise 1.5 (p.13). (a) `1100` (b) `0100` (c) `0100` (d) `1111`.

Solution to exercise 1.8 (p.13). To be a valid hypothesis, a decoded pattern must be a codeword of the code. If there were a decoded pattern in which the parity bits differed from the transmitted parity bits, but the source bits didn't differ, that would mean that there are two codewords with the same source bits but different parity bits. But since the parity bits are a deterministic function of the source bits, this is a contradiction.

So if any linear code is decoded with its optimal decoder, and a decoding error occurs anywhere in the block, some of the source bits must be in error.

Extra Solutions for Chapter 2

Solution to exercise 2.8 (p.30). Tips for Sketching the posteriors: best technique for sketching $p^{29}(1-p)^{271}$ is to sketch the logarithm of the posterior, differentiating to find where its maximum is. Take the second derivative at the maximum in order to approximate the peak as $\propto \exp[(p - p_{MAP})^2/2s^2]$ and find the width s.

Assuming the uniform prior (which of course is not fundamentally 'right' in any sense, indeed it doesn't look very uniform in other bases, such as the logit basis), the probability that the next outcome is a head is

$$\frac{n_H + 1}{N + 2} \tag{D.1}$$

(a) $N = 3$ and $n_H = 0$: $\frac{1}{5}$;

(b) $N = 3$ and $n_H = 2$: $\frac{3}{5}$;

(c) $N = 10$ and $n_H = 3$: $\frac{4}{12}$;

(d) $N = 300$ and $n_H = 29$: $\frac{30}{302}$.

Solution to exercise 2.27 (p.37). Define, for each $i > 1$, $p_i^* = p_i/(1 - p_1)$.

$$H(\mathbf{p}) = p_1 \log 1/p_1 + \sum_{i>1} p_i \log 1/p_i \quad \text{(D.2)}$$

$$= p_1 \log 1/p_1 + (1 - p_1) \sum_{i>1} p_i^* [\log 1/(1 - p_1) + \log 1/p_i^*] \quad \text{(D.3)}$$

$$= p_1 \log 1/p_1 + (1-p_1) \log 1/(1-p_1) + (1-p_1) \sum_{i>1} p_i^* [\log 1/p_i^*] \quad \text{(D.4)}$$

Similar approach for the more general formula.

Solution to exercise 2.28 (p.38). $P(0) = fg$; $P(1) = f(1-g)$; $P(2) = (1-f)h$; $P(3) = (1-f)(1-h)$; $H(X) = H_2(f) + fH_2(g) + (1-f)H_2(h)$. $\mathrm{d}H(X)/\mathrm{d}f = \log[(1-f)/f] + H_2(g) - H_2(h)$.

Solution to exercise 2.29 (p.38). Direct solution: $H(X) = \sum_i p_i \log 1/p_i = \sum_{i=1}^{\infty} (1/2^i) i = 2$. [The final step, summing the series, requires mathematical skill, or a computer algebra system; one strategy is to define $Z(\beta) = \sum_{i=1}^{\infty} (1/2^{\beta i})$, a series that is easier to sum (it's $Z = 1/(2^\beta - 1)$), then differentiate $\log Z$ with respect to β, evaluating at $\beta = 1$.]

Solution using decomposition: the entropy of the string of outcomes, H, is the entropy of the first outcome, plus (1/2)(the entropy of the remaining outcomes, assuming the first is a tail). The final expression in parentheses is identical to H. So $H = H_2(1/2) + (1/2)H$. Rearranging, $(1/2)H = 1$ implies $H = 2$.

Solution to exercise 2.30 (p.38). $P(\text{first is white}) = w/(w+b)$.

$P(\text{first is white, second is white}) = \frac{w}{w+b}\frac{w-1}{w+b-1}$.

$P(\text{first is black, second is white}) = \frac{b}{w+b}\frac{w}{w+b-1}$.

Now use the sum rule:

$P(\text{second is white}) = \frac{w}{w+b}\frac{w-1}{w+b-1} + \frac{b}{w+b}\frac{w}{w+b-1} = \frac{w(w-1)+bw}{(w+b)(w+b-1)} = \frac{w}{(w+b)}$.

Solution to exercise 2.31 (p.38). The circle lies in a square if the centre of the circle is in a smaller square of size $b - a$. The probability distribution of the centre of the circle is uniform over the plane, and these smaller squares make up a fraction $(b-a)^2/b^2$ of the plane, so this is the probability required. $(b-a)^2/b^2 = (1 - a/b)^2$.

Solution to exercise 2.32 (p.38). Buffon's needle. The angle t of the needle relative to the parallel lines is chosen at random. Once the angle is chosen, there is a probability $a \sin t/b$ that the needle crosses a line, since the distance between crossings of the parallel lines by the line aligned with the needle is $b/\sin t$. So the probability of crossing is $\int_{t=0}^{\pi/2} \mathrm{d}t\, a \sin t/b / \int_{t=0}^{\pi/2} dt = a/b[-\cos t]_0^{\pi/2}/(\pi/2) = (2/\pi)(a/b)$.

Solution to exercise 2.33 (p.38). Let the three segments have lengths x, y, and z. If $x + y > z$, and $x + z > y$, and $y + z > x$, then they can form a triangle. Now let the two points be located at a and b with $b > a$, and define $x = a$, $y = b - a$, and $z = 1 - b$. Then the three constraints imply

$b > 1 - b \;\Rightarrow\; b > 1/2$, similarly $a < 1/2$, and $b - a < 1/2$. Plotting these regions in the permitted (a, b) plane, we find that the three constraints are satisfied in a triangular region of area $1/4$ of the full area ($a > 0, b > 0, b > a$), so the probability is $1/4$.

Solution to exercise 2.36 (p.39). Assuming ignorance about the order of the ages F, A, and B, the six possible hypotheses have equal probability. The probability that $F > B$ is $1/2$.

The conditional probability that $F > B$ given that $F > A$ is given by the joint probability divided by the marginal probability:

$$P(F > B | F > A) = \frac{P(F > B, F > A)}{P(F > A)} = \frac{2/6}{1/2} = \frac{2}{3}. \tag{D.5}$$

(The joint probability that $F > B$ and $F > A$ is the probability that Fred is the oldest, which is $1/3$.)

Solution to exercise 2.37 (p.39). $1/5$.

Solution to exercise 2.39 (p.40). 9.716 bits.

Extra Solutions for Chapter 3

Solution to exercise 3.6 (p.54). The idea that complex models can win (in log evidence) by an amount linear in the number of data, F, and can lose by only a logarithmic amount is important and general.

For the biggest win by $\mathcal{H}_1$, let $F_\mathtt{a} = F$ and $F_\mathtt{b} = 0$.

$$\log \frac{P(\mathbf{s} \,|\, F, \mathcal{H}_1)}{P(\mathbf{s} \,|\, F, \mathcal{H}_0)} = \log \frac{1/F + 1}{p_0^F} = -\log(F + 1) + F \log 1/p_0. \tag{D.6}$$

The second term dominates, and the win for $\mathcal{H}_1$ is growing linearly with F.

For the biggest win by $\mathcal{H}_0$, let $F_\mathtt{a} = p_0 F$ and $F_\mathtt{b} = (1 - p_0)F$. We now need to use an accurate version of Stirling's approximation (1.17), because things are very close. The difference comes down to the square root terms in Stirling.

$$\begin{aligned}
\log \frac{P(\mathbf{s} \,|\, F, \mathcal{H}_1)}{P(\mathbf{s} \,|\, F, \mathcal{H}_0)} &= \log \tfrac{F_\mathtt{a}! F_\mathtt{b}!}{(F_\mathtt{a} + F_\mathtt{b} + 1)!} / p_0^{F_\mathtt{a}} (1 - p_0)^{F_\mathtt{b}} && \text{(D.7)} \\
&= \log(1/F + 1) - \log \binom{F}{F_\mathtt{a}} - \log p_0^{p_0 F} p_1^{p_1 F} && \text{(D.8)} \\
&= -\log(F + 1) + \frac{1}{2} \log \left[2\pi F \, \frac{p_0 F}{F} \, \frac{p_1 F}{F} \right] && \text{(D.9)} \\
&= -\frac{1}{2} \log \left[(F + 1) \left(1 + \frac{1}{F} \right) \right] + \frac{1}{2} \log \left[2\pi \, p_0 \, p_1 \right]. && \text{(D.10)}
\end{aligned}$$

Of these two terms, the second is asymptotically independent of F, and the first grows as half the logarithm of F.

Solution to exercise 3.10 (p.57). Let the variables be l, m, n, denoting the sex of the child who lives behind each of the three doors, with $l = 0$ meaning the first child is male. We'll assume the prior distribution is uniform, $P(l, m, n) = (1/2)^3$, over all eight possibilities. (Strictly, this is not a perfect assumption, since genetic causes do sometimes lead to some parents producing only one sex or the other.)

The first data item establishes that $l = 1$; the second item establishes that at least one of the three propositions $l = 0$, $m = 0$, and $n = 0$ is true.

The viable hypotheses are

$$l = 1,\ m = 0,\ n = 0;$$
$$l = 1,\ m = 1,\ n = 0;$$
$$l = 1,\ m = 0,\ n = 1.$$

These had equal prior probability. The posterior probability that there are two boys and one girl is 2/3.

Solution to exercise 3.12 (p.58). There are two hypotheses: let $H = 0$ mean that the original counter in the bag was white and $H = 1$ that is was black. Assume the prior probabilities are equal. The data is that when a randomly selected counter was drawn from the bag, which contained a white one and the unknown one, it turned out to be white. The probability of this result according to each hypothesis is:

$$P(D|H=0) = 1;\ P(D|H=1) = 1/2. \tag{D.11}$$

So by Bayes' theorem, the posterior probability of H is

$$P(H=0|D) = 2/3;\ P(H=1|D) = 1/3. \tag{D.12}$$

Solution to exercise 3.14 (p.58). It's safest to enumerate all four possibilities. Call the four equiprobable outcomes HH, HT, TH, TT. In the first three cases, Fred will declare he has won; in the first case, HH, whichever coin he points to, the other is a head; in the second and third cases, the other coin is a tail. So there is a 1/3 probability that 'the other coin' is a head.

Extra Solutions for Chapter 4

Solution to exercise 4.2 (p.68).

$$H(X,Y) = \sum_{x,y} P(x,y)h(x,y) = \sum_{x,y} P(x,y)(h(x)+h(y)) \tag{D.13}$$
$$= \left[\sum_{x,y} P(x,y)h(x)\right] + \left[\sum_{x,y} P(x,y)h(y)\right]. \tag{D.14}$$

Because $h(x)$ has no dependence on y, it's easy to sum over y in the first term. $\sum_y P(x,y) = P(x)$. Summing over x in the second term similarly, we have

$$H(X,Y) = \sum_x P(x)h(x) + \sum_y P(y)h(y) = H(X) + H(Y).$$

Solution to exercise 4.9 (p.84). If six are weighed against six, then the first weighing conveys no information about the question 'which is the odd ball?' All 12 balls are equally likely, both before and after.

If six are weighed against six, then the first weighing conveys exactly one bit of information about the question 'which is the odd ball and is it heavy or light?' There are 24 viable hypotheses before, all equally likely; and after, there are 12. A halving of the number of (equiprobable) possibilities corresponds to gaining one bit. (Think of playing `sixty-three`.)

Solution to exercise 4.10 (p.84). Let's use our rule of thumb: always maximize the entropy. At the first step we weigh 13 against 13, since that maximizes the entropy of the outcome. If they balance, we weigh 5 of the remainder against 4 of the remainder (plus one good ball). The outcomes have probabilities 8/26 (balance), 9/26, and 9/26, which is the most uniform distribution possible.

Let's imagine that the '5' are heavier than the '4 plus 1'. We now ensure that the next weighing has probability 1/3 for each outcome: leave out any three of the nine suspects, and allocate the others appropriately. For example, leaving out HHH, weigh HLL against HLL, where H denotes a possibly heavy ball and L a possibly light one. Then if those balance, weigh an omitted pair of H's; if they do not balance, weigh the two L's against each other.

John Conway's solution on page 86 of the book gives an explicit and more general solution.

Solution to exercise 4.11 (p.84). Going by the rule of thumb that the most efficient strategy is the most informative strategy, in the sense of having all possible outcomes as near as possible to equiprobable, we want the first weighing to have outcomes 'the two sides balance' in eight cases and 'the two sides do not balance' in eight cases. This is achieved by initially weighing 1,2,3,4 against 5,6,7,8, leaving the other eight balls aside. Iterating this binary division of the possibilities, we arrive at a strategy requiring 4 weighings.

The above strategy for designing a sequence of binary experiments by constructing a binary tree from the top down is actually not always optimal; the optimal method of constructing a binary tree will be explained in the next chapter.

Solution to exercise 4.12 (p.84). The weights needed are 1, 3, 9, and 27. Four weights in total. The set of 81 integers from -40 to $+40$ can be represented in ternary, with the three symbols being interpreted as 'weight on left', 'weight omitted', and 'weight on right'.

Solution to exercise 4.14 (p.84).

(a) A sloppy answer to this question counts the number of possible states, $\binom{12}{2}2^2 = 264$, and takes its base 3 logarithm, which is 5.07, which exceeds 5. We might estimate that six weighings suffice to find the state of the two odd balls among 12. If there are three odd balls then there are $\binom{12}{3}2^3 = 1760$ states, whose logarithm is 6.80, so seven weighings might be estimated to suffice.

However, these answers neglect the possibility that we will learn something more from our experiments than just which are the odd balls. Let us define the oddness of an odd ball to be the absolute value of the difference between its weight and the regular weight. There is a good chance that we will also learn something about the relative oddnesses of the two odd balls. If balls m and n are the odd balls, there is a good chance that the optimal weighing strategy will at some point put ball m on one side of the balance and ball n on the other, along with a load of regular balls; if m and n are both heavy balls, say, the outcome of this weighing will reveal, at the end of the day, whether m was heavier than n, or lighter, or the same, which is not something we were asked to find out. From the point of view of the task, finding the relative oddnesses of the two balls is a waste of experimental capacity.

A more careful estimate takes this annoying possibility into account.

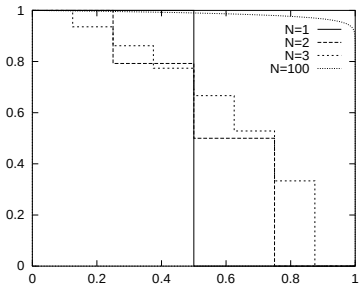
In the case of two odd balls, a complete description of the balls, including a ranking of their oddnesses, has three times as many states as we counted above (the two odd balls could be odd by the same amount, or by amounts that differ), i.e., $264 \times 3 = 792$ outcomes, whose logarithm is 6.07. Thus to identify the *full* state of the system in 6 weighings is

impossible – at least seven are needed. I don't know whether the original problem can be solved in 6 weighings.

In the case of three odd balls, there are $3! = 6$ possible rankings of the oddnesses if the oddnesses are different (e.g., $0 < A < B < C$), six if two of them are equal (e.g., $0 < A < B = C$ and $0 < A = B < C$), and just one if they are equal ($0 < A = B = C$). So we have to multiply the sloppy answer by 13. We thus find that the number of *full* system states is 13×1760, whose logarithm is 9.13. So at least ten weighings are needed to guarantee identification of the full state. I can believe that nine weighings might suffice to solve the required problem, but it is not clear.

(b) If the weights of heavy, regular and light balls are known in advance, the original sloppy method becomes correct. At least six weighings are needed to guarantee identification of the two-odd-out-of-twelve, and at least seven to identify three out of twelve.

Solution to exercise 4.16 (p.85). The curves $\frac{1}{N}H_\delta(Y^N)$ as a function of δ for $N = 1, 2, 3$ and 100 are shown in figure D.1. Note that $H_2(0.5) = 1$ bit.



$N = 2$		
δ	$\frac{1}{N}H_\delta(\mathbf{Y})$	$2^{H_\delta(\mathbf{Y})}$
0–0.25	1	4
0.25–0.5	0.79248	3
0.5–0.75	0.5	2
0.75–1	0	1

$N = 3$		
δ	$\frac{1}{N}H_\delta(\mathbf{Y})$	$2^{H_\delta(\mathbf{Y})}$
0–0.125	1	8
0.125–0.25	0.93578	7
0.25–0.375	0.86165	6
0.375–0.5	0.77398	5
0.5–0.625	0.66667	4
0.625–0.75	0.52832	3
0.75–0.875	0.33333	2
0.875–1	0	1

Figure D.1. $\frac{1}{N}H_\delta(\mathbf{Y})$ (vertical axis) against δ (horizontal), for $N = 1, 2, 3, 100$ binary variables with $p_1 = 0.5$.

Solution to exercise 4.19 (p.85). Chernoff bound. Let $t = \exp(sx)$ and $\alpha = \exp(sa)$. If we assume $s > 0$ then $x \geq a$ implies $t \geq \alpha$.

Assuming $s > 0$, $P(x \geq a) = P(t \geq \alpha) \leq \bar{t}/\alpha = \sum_x P(x)\exp(sx)/\exp(sa) = e^{-sa}g(s)$.

Changing the sign of s means that instead $x \leq a$ implies $t \geq \alpha$; so assuming $s < 0$, $P(x \leq a) = P(t \geq \alpha)$; the remainder of the calculation is as above.

Extra Solutions for Chapter 5

Solution to exercise 5.19 (p.102). The code $\{00, 11, 0101, 111, 1010, 100100, 0110\}$ is not uniquely decodeable because 11111 can be realized from $c(2)c(4)$ and $c(4)c(2)$.

Solution to exercise 5.20 (p.102). The ternary code $\{00, 012, 0110, 0112, 100, 201, 212, 22\}$ *is* uniquely decodeable because it is a prefix code.

Solution to exercise 5.23 (p.102). Probability vectors leading to a free choice in the Huffman coding algorithm satisfy $p_1 \geq p_2 \geq p_3 \geq p_4 \geq 0$ and

$$p_1 = p_3 + p_4. \tag{D.15}$$

The convex hull of $\mathcal{Q}$ is most easily obtained by turning two of the three inequalities $p_1 \geq p_2 \geq p_3 \geq p_4$ into equalities, and then solving equation (D.15) for $\mathbf{p}$. Each choice of equalities gives rise to one of the set of three vectors

$$\{1/3, 1/3, 1/6, 1/6\},\ \{2/5, 1/5, 1/5, 1/5\} \text{ and } \{1/3, 1/3, 1/3, 0\}. \tag{D.16}$$

Solution to exercise 5.24 (p.103). An optimal strategy asks questions that have a 50:50 chance of being answered yes or no. An essay on this topic should discuss practical ways of approaching this ideal.

Solution to exercise 5.25 (p.103). Let's work out the optimal codelengths. They are all integers. Now, the question is, can a set of integers satisfying the Kraft equality be arranged in an appropriate binary tree? We can do this constructively by going to the codeword supermarket and buying the shortest codewords first. Having bought them in order, they must define a binary tree.

Solution to exercise 5.27 (p.103).

a_i	p_i	$\log_2 \frac{1}{p_i}$	l_i	$c(a_i)$
a	0.09091	3.5	4	0000
b	0.09091	3.5	4	0001
c	0.09091	3.5	4	0100
d	0.09091	3.5	4	0101
e	0.09091	3.5	4	0110
f	0.09091	3.5	4	0111
g	0.09091	3.5	3	100
h	0.09091	3.5	3	101
i	0.09091	3.5	3	110
j	0.09091	3.5	3	111
k	0.09091	3.5	3	001

The entropy is $\log_2 11 = 3.4594$ and the expected length is $L = 3 \times \frac{5}{11} + 4 \times \frac{6}{11}$ which is $3\frac{6}{11} = 3.54545$.

Solution to exercise 5.28 (p.103). The key steps in this exercise are all spelled out in the problem statement. Difficulties arise with these concepts: (1) When you run the Huffman algorithm, all these equiprobable symbols will end up having one of just two lengths, $l^+ = \lceil \log_2 I \rceil$ and $l^- = \lfloor \log_2 I \rfloor$. The steps up to (5.32) then involve working out how many have each of these two adjacent lengths, which depends on how close I is to a power of 2. (2) The excess length was only defined for integer I, but we are free to find the maximum value is attains for any real I; this maximum will certainly not be exceeded by any integer I.

Solution to exercise 5.29 (p.103). The sparse source $\mathcal{P}_X = \{0.99, 0.01\}$ could be compressed with a Huffman code based on blocks of length N, but N would need to be quite large for the code to be efficient. The probability of the all-0 sequence of length N has to be reduced to about 0.5 or smaller for the code to be efficient. This sets $N \simeq \log 0.5 / \log 0.99 = 69$. The Huffman code would then have 2^{69} entries in its tree, which probably exceeds the memory capacity of all the computers in this universe and several others.

There are other ways that we could describe the data stream. One is run-length encoding. We could chop the source into the substrings $1, 01, 001, 0001, 00001, \ldots$ with the last elements in the set being, say, two strings of equal maximum length $00\ldots01$ and $00\ldots00$. We can give names to each of these strings and

compute their probabilities, which are not hugely dissimilar to each other. This list of probabilities starts $\{0.01, 0.0099, 0.009801, \ldots\}$. For this code to be efficient, the string with largest probability should have probability about 0.5 or smaller; this means that we would make a code out of about 69 such strings. It is perfectly feasible to make such a code. The only difficulty with this code is the issue of termination. If a sparse file ends with a string of 20 0s still left to transmit, what do we do? This problem has arisen because we failed to include the end-of-file character in our source alphabet. The best solution to this problem is to use an arithmetic code as described in the next chapter.

Solution to exercise 5.30 (p.103). The poisoned glass problem was intended to have the solution '129', this being the only number of the form $2^m + 1$ between 100 and 200. However the optimal strategy, assuming all glasses have equal probability, is to design a Huffman code for the glasses. This produces a binary tree in which each pair of branches have almost equal weight. On the first measurement, either 64 or 65 of the glasses are tested. (Given the assumption that one of the glasses is poisoned, it makes no difference which; however, going for 65 might be viewed as preferable if there were any uncertainty over this assumption.) There is a 2/129 probability that an extra test is needed after seven tests have occurred. So the expected number of tests is $7\frac{2}{129}$, whereas the strategy of the professor takes 8 tests with probability 128/129 and one test with probability 1/129, giving a mean number of tests $7\frac{122}{129}$. The expected waste is 40/43 tests.

Extra Solutions for Chapter 6

Solution to exercise 6.2 (p.117). Let's assume there are 128 viable ASCII characters. Then the Huffman method has to start by communicating 128 integers, each of which could in principle be as large as 127 or as small as 1, but plausible values will range from 2 to 17. There are correlations among these integers: if one of them is equal to 1, then none of the others can be 1. For practical purposes we might say that all the integers must be between 1 and 32 and use a binary code to represent them in 5 bits each. Then the header will have a size of $5 \times 128 = 640 bits$.

If the file to be compressed is short – 400 characters, say – then (taking 4 as a plausible entropy per character, if the frequencies are known) the compressed length would be 640 (header) + 1600 (body) $\simeq$ 2240, if the compression of the body is optimal. For any file much shorter than this, the header is clearly going to dominate the file length.

When we use the Laplace model, the probability distribution over characters starts out uniform and remains roughly so until roughly 128 characters have been read from the source. In contrast, the Dirichlet model with $\alpha = 0.01$ only requires about 2 characters to be read from the source for its predictions to be strongly swung in favour of those characters.

For sources that do use just a few characters with high probability, the Dirichlet model will be better. If actually all characters are used with near-equal probability then $\alpha = 1$ will do better.

The special case of a large file made entirely of equiprobable 0s and 1s is interesting. The Huffman algorithm has to assign codewords to all the other characters. It will assign one of the two used characters a codeword of length 1, and the other gets length 2. The expected filelength is thus more than $(3/2)N$, where N is the source file length. The arithmetic codes will give an expected filelength that asymptotically is $\sim N$.

It is also interesting to talk through the case where one character has huge probability, say 0.995. Here, the arithmetic codes give a filelength that's asymptotically less than N, and the Huffman method tends to N from above.

Solution to exercise 6.4 (p.119). Assume a code maps all strings onto strings of the same length or shorter. Let L be the length of the *shortest* string that is made shorter by this alleged code, and let that string be mapped to an output string of length l. Take the set of all input strings of length less than or equal to l, and count them. Let's say there are $n^{\text{in}}(l)$ of length l. [$n^{\text{in}}(l) = A^l$, where A is the alphabet size.]

Now, how many output strings of length l do these strings generate? Well, for any length $< L$, by the definition of L, all the input strings were mapped to strings with the same length. So the total number of output strings of length l must be at least $n^{\text{in}}(l) + 1$, since not only all the inputs of length l, but also the 'shortest' input string, defined above, maps to an output of length l.

By the pigeonhole principle, that's too many pigeons for the available holes. Two of those output strings must be identical, so the mapping is not uniquely decodeable.

Solution to exercise 6.7 (p.123). Figure D.2 shows the left-hand side of the arithmetic encoder for the case $N = 5$, $K = 2$.

0	0	0	1	1
		1	0	1
			1	0
	1	0	0	1
			1	0
		1	0	0
1	0	0	0	1
			1	0
		1	0	0
	1	0	0	0

Figure D.2. Arithmetic encoder for binary strings of length $N = 5$ with fixed weight $K = 2$. (The right-hand side, a regular binary scale, has been omitted.)

a_i	p_i	$h(p_i)$	l_i	$c(a_i)$
111	1e-6	19.9	5	00000
110	1e-4	13.3	5	00001
101	1e-4	13.3	5	00010
011	1e-4	13.3	5	00011
001	0.0098	6.7	3	001
010	0.0098	6.7	3	010
100	0.0098	6.7	3	011
000	0.97	0.0	1	1

Solution to exercise 6.9 (p.123). Using the Huffman algorithm we arrive at the symbol code shown in the margin. The expected length is roughly 1. The entropy of $\mathbf{x}$ is 0.24. The ratio length / entropy is 4, to 1 decimal place.

An arithmetic code for a string of length $N = 1000$, neglecting the termination overhead, gives an expected length equal to N times the entropy, i.e. 80 bits.

The variance of the length is found from the variance of the number of 1s, which is Npq; the length is linearly related to the number of 1s, r

$$l(r) = r \log \frac{1}{f_1} + (N - r) \log \frac{1}{f_0} = r \log \frac{f_0}{f_1} + N \log \frac{1}{f_0}, \tag{D.17}$$

so the standard deviation is $3.14 \log[f_0/f_1] = 21$. So the compressed length is expected to be 80 ± 21 bits. Or at most two more than this, allowing for worst-case termination.

Solution to exercise 6.10 (p.124). One can generate strings with density f by running dense random bits into the decoder corresponding to the arithmetic encoder for a sparse source with density f. See figure D.3.

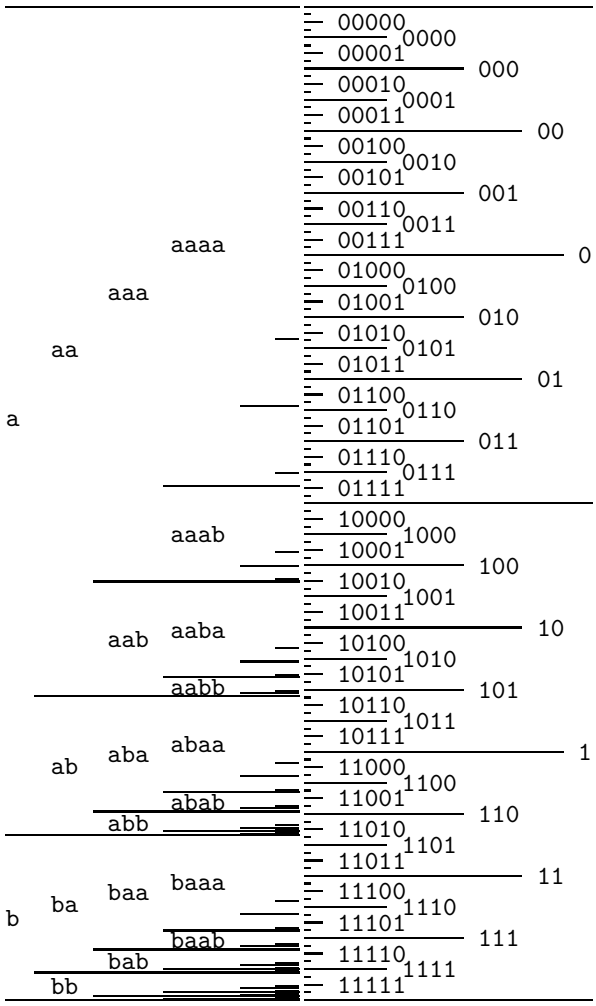


Figure D.3. Arithmetic encoder for a sparse source with $f = 1/6$.

Solution to exercise 6.11 (p.124). The encoding is 00**1**1010110**0**0000110100101000001**1**, coming from the following parsed message:

$$(,0), (0,\mathbf{1}), (1,0), (10,1), (10,\mathbf{0}), (00,0), (011,0), (100,1), (010,0), (001,\mathbf{1})$$

The highlighted symbols would be omitted in the further improved coding system.

a_i	p_i	$h(p_i)$	l_i	c_i
a	0.01	6.6	6	000000
b	0.02	5.6	6	000001
c	0.04	4.6	5	00001
d	0.05	4.3	4	0010
e	0.06	4.1	4	0011
f	0.08	3.6	4	0001
g	0.09	3.5	3	100
h	0.1	3.3	3	101
i	0.25	2.0	2	11
j	0.3	1.7	2	01

Extra Solutions for Chapter 6.8

Solution to exercise 6.15 (p.125). Using the Huffman coding algorithm, we arrive at the answer shown, which is unique (apart from trivial modifications to the codewords).

The expected length is 2.81. The entropy is 2.78.

Solution to exercise 6.16 (p.125). The entropy of $\mathbf{y} = x_1x_2$ is twice $H(X)$; $H(X) = 1.295$ bits so $H(\mathbf{Y}) = 2.59$ bits.

The optimal binary symbol code is constructed using the Huffman coding algorithm. There are several valid answers; the codelengths should be identical to one of the two lists below. The strings `ab` and `ba`, marked $\star$, are interchangeable.

a_i	p_i	$h(p_i)$	$l_i^{(1)}$	$c(a_i)$	$l_i^{(2)}$	$c^{(2)}(a_i)$
`aa`	0.01	6.6	6	`000000`	6	`000000`
`ab`*	0.03	5.1	6	`000001`	6	`000001`
`ba`*	0.03	5.1	5	`00001`	5	`00001`
`ac`	0.06	4.1	4	`0010`	4	`0010`
`ca`	0.06	4.1	4	`0011`	4	`0011`
`bb`	0.09	3.5	4	`0001`	4	`0001`
`bc`	0.18	2.5	3	`010`	2	`10`
`cb`	0.18	2.5	3	`011`	2	`11`
`cc`	0.36	1.5	1	`1`	2	`01`

The expected length is 2.67.

Solution to exercise 6.17 (p.125). 470 ± 30.

Solution to exercise 6.18 (p.125). Maximize $R = S/L = \sum p_n \log(1/p_n) / \sum p_n l_n$ subject to normalization. gives $(\mathrm{d}S/\mathrm{d}p_n L - \mathrm{d}L/\mathrm{d}p_n S)/L^2 = \mu$ gives $\mathrm{d}S/\mathrm{d}p_n = Rl_n + \mu L$, with $\mathrm{d}S/\mathrm{d}p_n = -1 - \log p_n$. Thus $p_n = \exp(-Rl_n)/Z$.

Notice that this distribution has two properties: $\mathrm{d}\log Z/\mathrm{d}R = -L$

$$S = \log Z + RL$$

$$S/L = \log Z/L + R$$

this instantly means $\log Z = 0$ without my having to do any differentiation!

Solution to exercise 6.19 (p.126). There are 52! orderings of a pack of cards, so the minimum number of bits required to make a perfect shuffle is $\log_2(52!) \simeq 226$ bits.

Solution to exercise 6.20 (p.126). (Draft solution, more below.)

(a) After the cards have been dealt, the number of bits needed for North to convey her hand to South (remember that he already knows his own hand) is

$$\log_2 \binom{39}{13} \simeq 33 \,\text{bits}. \tag{D.18}$$

Now, North does not know South's hand, so how, in practice, could this information be conveyed efficiently? [This relates to the Slepian-Wolf correlated information comunication problem.]

(b) The maximum number of bits is equal to 35, the number of distinct bids in the list $1\clubsuit \ldots 7NT$. Given the assumption that E and W do not bid, the bidding process can be viewed as defining a binary string of length 35, with a 1 against every bid that was made by N or S, and a 0 against every bid that was skipped. The complete bidding history can be reconstructed from this binary string, since N and S alternate (we assumed that the bidding stops if either of them does not bid). So the maximum total information conveyed cannot exceed 35 bits.

A bidding system that achieved this maximum information content would be one in which a binary code is agreed such that 0s and 1s are equally

probable; then each bidder chooses the next bid by raising the bid by the appropriate number of notches. There will be a probability of $1/2$ that they raise the bid by one notch; a probability of $1/4$ that they raise it by two notches; and so forth.

Solution to exercise 6.20 (p.126). (a) From the point of view of South, the information content of North's hand is $\log \binom{(52-13)}{13} \simeq 33$ bits.

(b) The list of bids not made and bids made forms a binary string. We could write the omitted bids as 0s and the made bids as 1s. Then North and South, by their raises, are defining a binary string of length 35. They can thus convey a total of at most 35 bits to each other. It's conceivable therefore that each could learn about half of the information content of the other's hand (33 bits).

Solution to exercise 6.21 (p.126). (First of two solutions.)

(a) The arabic keypad can produce the times 0:01–0:09 in two symbols and the times 0:10–0:59 in three symbols. The roman keypad can produce the times 0:01, 0:10, 1:00, and 10:00 in two symbols, and 0:02, 0:11, 0:20, 1:01, 1:10, 2:00, 20:00, 11:00, 10:10, and 10:01 in three symbols. The times 0:11, 1:01, 1:10, 11:00, 10:10, and 10:01 can all be produced in two different ways, because the two keys with numbers can be pressed in either sequence.

(b) The arabic code is incomplete because

i. The keys `0` and □ are both illegal first symbols.

ii. After a four-digit number has been entered, the only legal symbol is □.

The roman code is incomplete because

i. The key □ is an illegal first symbol.

ii. Some times can be produced by several symbol-strings. A time such as 1:23 can be entered as `CXXIII`□ or as `IICIXX`□.

iii. After a key has been pressed a number of times (five or nine, depending which key) it may not be pressed any more times.

(c) The arabic code can produce 3:15, 2:30, and 5:00 in four symbols, and the roman code cannot. The roman code can produce 12:00 and 21:00 in four symbols, and the arabic code cannot.

(d) Both codes allow the time 0:01 to be encoded with a very short sequence, length two. This is implicitly one of the most probable times for both models. In the arabic model, the implicit probability of a time is roughly $1/11^{l+1}$, where l is the length of the time when encoded in decimal. In the roman model, times that contain many ones and zeroes are the most probable, with the probability of a time decreasing roughly as the sum of its digits: $P(\mathbf{x}) \sim 1/5^s$, where $s = \sum_i x_i$.

(e) When I use the microwave, my probability distribution is roughly:

$\mathbf{x}$	0:10	0:20	0:30	1:00	1:10	1:20	1:30	2:00	3:00
$P(\mathbf{x})$	0.1	0.05	0.01	0.1	0.01	0.1	0.5	0.03	0.03

$\mathbf{x}$	5:00	7:00	8:00	10:00	12:00	20:00	other
$P(\mathbf{x})$	0.02	0.02	0.02	0.01	0.01	0.01	ϵ

The arabic system is poorly matched to my distribution because it forces me to push the zero button at the end of every time, to specify 'zero seconds', which I always want. The roman system similarly wastes an entire button (the `I` button) which I never use. The arabic system is otherwise quite well matched to my probability distribution, except that my favourite time (1:30 for a cafe latte) could do with a shorter sequence. The roman system is well–matched to some of my times, but terribly matched to others, in particular, the time 8:00.

The arabic system has a maximum codelength of five symbols. The roman system has a terrible maximum codelength of 28 symbols, for the time 59:59.

(f) An alternative encoder using five buttons would work as follows.

i. The display starts by offering as a default the median of the last one hundred times selected. If the user has a favourite time, then this will probably be offered. It can be accepted by pressing the □ key.

ii. The other four keys are marked +, ++, −, and −−.

- The + symbol increments the displayed time by a little bit, e.g.16%.
- The − symbol decrements the displayed time by a little bit, e.g.16%.
- The ++ symbol increments the displayed time by a lot, e.g., a factor of two.
- The −− symbol decrements the displayed time by a lot, e.g., a factor of two.

To make this system even more adaptive to its user, these four buttons could have their effect by moving the percentile around the distribution of times recently used by the user. The initial time is the median. The + button takes us to the 63rd percentile and ++ takes us to the 87th, say, with the step size decreasing adaptively as the time is selected. If a user has five preferred times, these could adaptively be discovered by the system so that, after time, the five times would be invoked by the sequences −−□, −□, □, +□, and ++□ respectively.

Solution to exercise 6.21 (p.126). (a) The arabic microwave can generate the times 0, 1, 2, ... 9 seconds with two symbols, and the times from 0 to 59 seconds (or perhaps 99 seconds) with three symbols. Not a very good use of the shortest strings!

The roman microwave can generate 1 second, 10 seconds, 1 minute, and 10 minutes with two symbols, and any of the 10 sums of those 4 quantities with three symbols. Again, not a good use of the shortest strings, except perhaps 1 minute and 10 minutes. Also there is a bit of inefficiency: the sequences CX and XC both produce 1 minute and 10 seconds.

(b) The codes are not complete. In a complete code, there would be a unique way to encode each cooking time, and there would be no redundant symbols (whereas all times in both codes end with "Start", which is in at least some cases redundant).

(c) 1 minute 23 seconds; 30 minutes.

(d) The implicit probability distribution over digits is uniform with arabic, and the distribution over the number of non-zero digits is an exponential, with

short strings (such as 8 seconds) being more probable than long ones (such as 10 minutes).

The implicit distribution over digits is highly nonuniform in the roman microwave. The digit 1 is much more probable than the digit 9.

For most large times (such as twenty minutes) there are intermediate and small times (such as twenty seconds, and two seconds) that have the same implicit probability. So the roman distribution, while rather strange, does have a scale-free property.

(e) My distribution over times has a lot of mass around 1:30, 2:00, and 4:00.

(f) No-one needs a ten-minute cooking event timed to within a second. An efficient system would have variable grain size, with 1-second precision at the ten-second end of things, 5-second precision in the 1-minute area, and twenty-second precision at ten minutes.

Extra Solutions for Chapter 7

Solution to exercise 7.2 (p.134).

- The self-delimiting representations of n are:

$$\begin{aligned} c_\alpha(n) = \; & \overline{000000000000000000000000000000000000000000000000000000000000} \\ & \overline{00000000000000000000000000000000000000} \\ & \overline{1}0000111101100110000111101011100100110010101000001010011110 \\ & 10001100001110111011011011011111101110. \end{aligned}$$

$$\begin{aligned} c_\beta(n) = \; & 000000 \\ & 1\overline{100010} \\ & 0000111101100110000111101011100100110010101000001010011110 \\ & 10001100001110111011011011011111101110. \end{aligned}$$

$$\begin{aligned} c_\gamma(n) = \; & \overline{00111} \\ & \overline{100010} \\ & 0000111101100110000111101011100100110010101000001010011110 \\ & 10001100001110111011011011011111101110. \end{aligned}$$

$$\begin{aligned} c_\delta(n) = \; & 01\overline{1} \\ & 11 \\ & \overline{100010} \\ & 0000111101100110000111101011100100110010101000001010011110 \\ & 10001100001110111011011011011111101110. \end{aligned}$$

- Byte based, base 3.

```
010010100110101010011010010110000010101001010101010110001010010
110011000010000010010100101000010101010010100101001100001100011
```

- base 7

```
011000101000101000100011100110010100001100000110011001101100000
100000100000010101000011001011100100010100111
```

- base 15

  100111101010010000100110010000101011100011101011101000001110111
  000111011101010101110011111101110100001111

Solution to exercise 7.3 (p.135). A code that has shorter codelengths asymptotically (e.g., for $n > 2^{100}$) uses the same idea but first encodes the number of levels of recursion that the encoder will go through, using any convenient prefix code for integers, for example C_ω; then the encoder can use $c_B(n)$ instead of $c_b(n)$ to encode each successive integer in the recursion, and can omit the terminal zero.

Extra Solutions for Chapter 8

Solution to exercise 8.1 (p.140).

$$H(X,Y) = H(U,V,V,W) = H(U,V,W) = H_u + H_v + H_w. \tag{D.19}$$

$$H(X|Y) = H_u. \tag{D.20}$$

$$I(X;Y) = H_v. \tag{D.21}$$

Solution to exercise 8.5 (p.140). The entropy distance:

$$D_H(X,Y) \equiv H(X,Y) - I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x)P(y)}{P(x,y)^2}. \tag{D.22}$$

is fairly easily shown to satisfy the first three axioms $D_H(X,Y) \geq 0$, $D_H(X,X) = 0$, $D_H(X,Y) = D_H(Y,X)$.

A proof that it obeys the triangle inequality is not so immediate. It helps to know in advance what the difference $D(X,Y) + D(Y,Z) - D(X,Z)$ should add up to; this is most easily seen by first making a picture in which the quantities $H(X), H(Y)$, and $H(Z)$ are represented by overlapping areas, c.f. figure 8.2 and exercise 8.8 (p.141). Such a picture indicates that $D(X,Y) + D(Y,Z) - D(X,Z) = H(Y|X,Z) + I(X;Z|Y)$.

$$\begin{aligned}
& D(X,Y) + D(Y,Z) - D(X,Z) \\
&= \sum_{x,y,z} P(x,y,z) \log \frac{P(x)P(y)P(y)P(z)P(xz)^2}{P(xy)^2 P(x)P(z)P(y,z)^2} && \text{(D.23)} \\
&= 2 \sum_{x,y,z} P(x,y,z) \log \frac{P(x,z)P(x,z|y)}{P(x,y,z)P(x|y)P(z|y)} && \text{(D.24)} \\
&= 2 \sum_{x,y,z} P(x,y,z) \left[\log \frac{1}{P(y|xz)} + \log \frac{P(x,z|y)}{P(x|y)P(z|y)} \right] && \text{(D.25)} \\
&= 2 \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log \frac{1}{P(y|x,z)} + && \text{(D.26)} \\
&\quad 2 \sum_{y} P(y) \sum_{x,z} P(x,z|y) \log \frac{P(x,z|y)}{P(x|y)P(z|y)} && \text{(D.27)} \\
&= 2 \sum_{x,z} P(x,z) H(Y|x,z) + 2 \sum_{y} P(y) I(X;Z|y). && \text{(D.28)} \\
&= 2H(Y|X,Z) + 2I(X;Z|Y). && \text{(D.29)}
\end{aligned}$$

The quantity $I(X;Z|Y)$ is a conditional mutual information, which like a mutual information is positive. The other term $H(Y|X,Z)$ is also positive, so $D(X,Y) + D(Y,Z) - D(X,Z) \geq 0$.

Solution to exercise 8.10 (p.142). Seeing the top of the card *does* convey information about the colour of its other side. Bayes' theorem allows us to draw the correct inference in any given case, and Shannon's mutual information is the measure of how much information is conveyed, on average.

This inference problem is equivalent to the three doors problem. One quick way to justify the answer without writing down Bayes' theorem is 'The probability that the lower face is opposite in colour to the upper face is always 1/3, since only one of the three cards has two opposite colours on it'.

The joint probability of the two colours is

$$\begin{array}{c|cc} P(u,l) & u=0 & u=1 \\ \hline l=0 & 1/3 & 1/6 \\ l=1 & 1/6 & 1/3 \end{array} \tag{D.30}$$

The marginal entropies are $H(U) = H(L) = 1\,\text{bit}$, and the mutual information is

$$I(U;L) = 1 - H_2(1/3) = 0.08\,\text{bits}. \tag{D.31}$$

Extra Solutions for Chapter 9

Solution to exercise 9.17 (p.155). The conditional entropy of Y given X is $H(Y|X) = \log 4$. The entropy of Y is at most $H(Y) = \log 10$, which is achieved by using a uniform input distribution. The capacity is therefore

$$C = \max_{\mathcal{P}_X} H(Y) - H(Y|X) = \log 10/4 = \log 5/2\,\text{bits}. \tag{D.32}$$

Solution to exercise 9.19 (p.156). The number of recognizable '2's is best estimated by concentrating on the type of patterns that make the greatest contribution to this sum. These are patterns in which just a small patch of the pixels make the shape of a 2 and most of the other pixels are set at random. It is unlikely that the random pixels will take on some other recognizable shape, as we will confirm later. A recognizable letter 2 surrounded by a white border can be written in 6×7 pixels. This leaves 214 pixels that can be set arbitrarily, and there are also 12×11 possible locations for the miniature 2 to be placed, and two colourings (white on black / black on white). There are thus about $12 \times 11 \times 2 \times 2^{214} \simeq 2^{219}$ miniature 2 patterns, almost all of which are recognizable only as the character 2. This claim that the noise pattern will not look like some other character is confirmed by noting what a small fraction of all possible patterns the above number of 2s is. Let's assume there are 127 other characters to worry about. Only a fraction 2^{-37} of the 2^{256} random patterns are recognizable 2s, so similarly, of the 2^{219} miniature 2 patterns identified above, only a fraction of about 127×2^{-37} of them also contain another recognizable character. These double-hits decrease undetectably the above answer, 2^{219}.

Figure D.4. Four random samples from the set of 2^{219} 'miniature 2s' defined in the text.

Another way of estimating the entropy of a 2, this time banning the option of including background noise, is to consider the number of *decisions* that are made in the construction of a font. A font may be **bold (2)** or not bold; *italic (2)* or not; sans-serif (2) or not. It may be normal size (2), small (2) or tiny (2). It may be calligraphic, futuristic, modern, or gothic. Most of these choices are independent. So we have at least $2^4 \times 3^2$ distinct fonts. I imagine that Donald Knuth's METAFONT, with the aid of which this document was produced, could turn each of these axes of variation into a continuum so that arbitrary intermediate fonts can also be created. If we can distinguish, say, five degrees of boldness, ten degrees of italicity, and so forth, then we can imagine creating perhaps $10^6 \simeq 2^{20}$ distinguishable fonts, each with a distinct 2. Extra parameters such as loopiness and spikiness could further increase this number. It would be interesting to know how many distinct 2s METAFONT can actually produce in a 16×16 box.

The entropy of the probability distribution $P(y \mid x = 2)$ depends on the assumptions about noise and character size. If we assume that noise is unlikely, then the entropy may be roughly equal to the number of bits to make a clean 2 as discussed above. The possibility of noise increases the entropy. The largest it could plausibly be is the logarithm of the number derived above for the number of patterns that are recognizable as a 2, though I suppose one could argue that when someone writes a 2, they may end up producing a pattern $\mathbf{y}$ that is not recognizable as a 2. So the entropy could be even larger than 220 bits. It should be noted however, that if there is a 90% chance that the 2 is a clean 2, with entropy 20 bits, and only a 10% chance that it is a miniature 2 with noise, with entropy 220 bits, then the entropy of y is $H_2(0.1) + 0.1 \times 220 + 0.9 \times 20 \simeq 40$ bits, so the entropy would be much smaller than 220 bits.

Solution to exercise 9.21 (p.156). The probability of error is the probability that the selected message is not uniquely decodeable by the receiver, i.e., it is the probability that one or more of the $S-1$ other people has the same birthday as our selected person, which is

$$1 - \left(\frac{A-1}{A}\right)^{S-1} \;=\; 1 - 0.939 \;=\; 0.061. \tag{D.33}$$

The capacity of the communication channel is $\log 365 \simeq 8.5$ bits. The rate of communication attempted is $\log 24 \simeq 4.6$ bits.

So we are transmitting substantially below the capacity of this noiseless channel, and our communication scheme has an appreciable probability of error (6%). Random coding looks a rather silly idea.

Solution to exercise 9.22 (p.157). The number of possible K-tuples is A^K, and we select q^K such K-tuples, where q is the number of people in each of the K rooms. The probability of error is the probability that the selected message is not uniquely decodeable by the receiver,

$$1 - \left(\frac{A^K - 1}{A^K} \right)^{q^K - 1} . \tag{D.34}$$

In the case $q = 364$ and $K = 1$ this probability of error is

$$1 - \left(1 - \frac{1}{A} \right)^{q-1} \simeq 1 - e^{-(q-1)/A} \simeq 1 - e = 0.63. \tag{D.35}$$

[The exact answer found from equation (D.34) is 0.631.] Thus random coding is highly likely to lead to a communication failure.

As K gets large, however, we can approximate

$$1 - \left(\frac{A^K - 1}{A^K} \right)^{q^K - 1} = 1 - \left(1 - \frac{1}{A^K} \right)^{q^K - 1} \simeq \frac{q^K - 1}{A^K} \simeq \left(\frac{q}{A} \right)^K . \tag{D.36}$$

In the example $q = 364$ and $A = 365$, this probability of error decreases as $10^{-0.0012K}$, so, for example, if $K \simeq 6000$ then the probability of error is smaller than 10^{-6}.

For sufficiently large blocklength K, random coding becomes a reliable, albeit bizarre, coding strategy.

Extra Solutions for Chapter 10

Solution to exercise 10.1 (p.168). Consider a string of bit pairs $b_k, \hat{b}_k$, having the property that $\sum_{k=1}^{K} P(\hat{b}_k \neq b_k)/K = p_{\rm b}$. These bits are concatenated in blocks of size $K = NR$ to define the quantities s and $\hat{s}$. Also, $P(b_k = 1) = 1/2$. We wish to show that these properties imply $I(s; \hat{s}) \geq K(1 - H_2(p_{\rm b}))$, regardless of whether there are correlations among the bit errors.

More to come here.

Solution to exercise 10.12 (p.172). $I(X;Y) = H(X) - H(X|Y)$.

$I(X;Y) = H_2(p_0) - qH_2(p_0)$.

Maximize over p_0, get $C = 1 - q$.

The $(2, 1)$ code is $\{\texttt{01}, \texttt{10}\}$. With probability q, the `1` is lost, giving the output `00`, which is equivalent to the "`?`" output of the Binary Erasure Channel. With probability $(1 - q)$ there is no error; the two input words and the same two output words are identified with the `0` and `1` of the BEC. The equivalent BEC has erasure probability q. Now, this shows the capacity of the Z channel is at least half that of the BEC.

This result is a bound, not an inequality, because our code constrains the input distribution to be 50:50, which is not necessarily optimal, and because we've introduced simple anticorrelations among successive bits, which optimal codes for the channel would not do.

Extra Solutions for Chapter 11

Solution to exercise 11.3 (p.184). In a nutshell, the encoding operations involve 'additions' and 'multiplies', and these operations are associative.

Let the source block be $\{s_{k_2k_1}\}$ and the transmitted block be $\{t_{n_2n_1}\}$. Let the two generator matrices be $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$. To conform to convention, these matrices have to be transposed if they are to right-multiply.

If we encode horizontally first, then the intermediate vector is

$$u_{k_2n_1} = \sum_{k_1} G^{(1)\mathsf{T}}_{n_1k_1} s_{k_2k_1} \tag{D.37}$$

and the transmission is

$$\begin{aligned} t_{n_2n_1} &= \sum_{k_2} G^{(2)\mathsf{T}}_{n_2k_2} u_{k_2n_1} &\text{(D.38)}\\ &= \sum_{k_2} G^{(2)\mathsf{T}}_{n_2k_2} \sum_{k_1} G^{(1)\mathsf{T}}_{n_1k_1} s_{k_2k_1}. &\text{(D.39)} \end{aligned}$$

Now, by the associative property of addition and multiplication, we can reorder the summations and multiplications:

$$\begin{aligned} t_{n_2n_1} &= \sum_{k_1}\sum_{k_2} G^{(2)\mathsf{T}}_{n_2k_2} G^{(1)\mathsf{T}}_{n_1k_1} s_{k_2k_1} &\text{(D.40)}\\ &= \sum_{k_1} G^{(1)\mathsf{T}}_{n_1k_1} \sum_{k_2} G^{(2)\mathsf{T}}_{n_2k_2} s_{k_2k_1}. &\text{(D.41)} \end{aligned}$$

This is identical to what happens if we encode vertically first, getting intermediate vector

$$v_{n_2k_1} = \sum_{k_2} G^{(2)\mathsf{T}}_{n_2k_2} s_{k_2k_1} \tag{D.42}$$

then transmitting

$$t_{n_2n_1} = \sum_{k_1} G^{(1)\mathsf{T}}_{n_1k_1} v_{n_2k_1}. \tag{D.43}$$

Solution to exercise 11.6 (p.188). The fraction of all codes that are linear is absolutely tiny. We can estimate the fraction by counting how many linear codes there are and how many codes in total.

A linear (N, K) code can be defined by the $M = N - K$ constraints that it satisfies. The constraints can be defined by a $M \times N$ parity-check matrix. Let's count how many parity-check matrices there are, then correct for overcounting in a moment. There are 2^{MN} distinct parity-check matrices. Most of these have nearly full rank. If the rows of the matrix are rearranged, that makes no difference to the code. Indeed, you can multiply the matrix $\mathbf{H}$ by any square invertible matrix, and there is no change to the code. Row-permutation is a special case of multiplication by a square matrix. So the size of the equivalence classes of parity-check matrix is 2^{M^2}. (For every parity-check matrix, there are 2^{M^2} ways of expressing it.) So the number of different linear codes is $2^{MN}/2^{M^2} = 2^{MK}$.

The total number of codes is the number of choices of 2^K words from the set of 2^N possible words, which is $\binom{2^N}{2^K}$, which is approximately

$$\frac{(2^N)^{2^K}}{(2^K)!} = \frac{2^{N2^K}}{(2^K)!}. \tag{D.44}$$

The fraction required is thus

$$\frac{2^{N^2R(1-R)}(2^K)!}{2^{N2^K}}. \tag{D.45}$$

Solution to exercise 11.8 (p.188). A code over $GF(8)$ We can denote the elements of $GF(8)$ by $\{0, 1, A, B, C, D, E, F\}$. Each element can be mapped onto a polynomial over $GF(2)$.

element	polynomial	binary representation
0	0	000
1	1	001
A	x	010
B	$x+1$	011
C	x^2	100
D	x^2+1	101
E	x^2+x	110
F	x^2+x+1	111

(D.46)

The multiplication and addition operations are given by multiplication and addition of the polynomials, modulo $x^3 + x + 1$.

Here is the multiplication table:

$\cdot$	0	1	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0
1	0	1	A	B	C	D	E	F
A	0	A	C	E	B	1	F	D
B	0	B	E	D	F	C	1	A
C	0	C	B	F	E	A	D	1
D	0	D	1	C	A	F	B	E
E	0	E	F	1	D	B	A	C
F	0	F	D	A	1	E	C	B

(D.47)

Here is a (9,2) code over $GF(8)$ generated by the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & A & B & C & D & E & F \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{D.48}$$

```
000000000   011111111   0AAAAAAAA   0BBBBBBBB   0CCCCCCCC   0DDDDDDDD   0EEEEEEEE   0FFFFFFFF
101ABCDEF   110BADCFE   1AB01EFCD   1BA10FEDC   1CDEF01AB   1DCFE10BA   1EFCDAB01   1FEDCBA10
A0ACEB1FD   A1BDFA0EC   AA0EC1BDF   AB1FD0ACE   ACE0AFDB1   ADF1BECA0   AECA0DF1B   AFDB1CE0A
B0BEDFC1A   B1AFCED0B   BA1CFDEB0   BB0DECFA1   BCFA1B0DE   BDEB0A1CF   BED0B1AFC   BFC1A0BED
C0CBFEAD1   C1DAEFBC0   CAE1DC0FB   CBF0CD1EA   CC0FBAE1D   CD1EABF0C   CEAD10CBF   CFBC01DAE
D0D1CAFBE   D1C0DBEAF   DAFBE0D1C   DBEAF1C0D   DC1D0EBFA   DD0C1FAEB   DEBFAC1D0   DFAEBD0C1
E0EF1DBAC   E1FE0CABD   EACDBF10E   EBDCAE01F   ECABD1FE0   EDBAC0EF1   EE01FBDCA   EF10EACDB
F0FDA1ECB   F1ECB0FDA   FADF0BCE1   FBCE1ADF0   FCB1EDA0F   FDA0FCB1E   FE1BCF0AD   FF0ADE1BC
```

Further exercises that can be based on this example:

▷ Exercise D.1.[2] Is this code a perfect code?

▷ Exercise D.2.[2] Is this code a maximum distance separable code?

Extra Solutions

Solution to exercise D.1 (p.720). The $(9, 2)$ code has $M = 7$ parity checks, and its distance is $d = 8$. If the code were perfect, then all points would be at a distance of at most $d/2$ from the nearest codeword, and each point would only have one nearest codeword.

The $(9, 2)$ code is not a perfect code. Any code with even distance cannot be a perfect code because it must have vectors that are equidistant from the two nearest codewords, for example, `000001111` is at Hamming distance 4 from both `000000000` and `011111111`.

We can also find words that are at a distance greater than $d/2$ from all codewords, for example `111110000`, which is at a distance of five or more from all codewords.

Solution to exercise D.2 (p.720). The $(9, 2)$ code is maximum distance separable. It has $M = 7$ parity checks, and when any 7 characters in a codeword are erased we can restore the others. Proof: any two by two submatrix of $\mathbf{G}$ is invertible.

Extra Solutions for Chapter 12

Solution to exercise 12.9 (p.201). $\log_{36} 6,000,000,000 = 6.3$, so a 7-character address could suffice, if we had no redundancy. One useful internet service provided by shortURL.com is the service of turning huge long URLs into tiny ones, using the above principle.

Email addresses can be as short as four characters (I know m@tc), but roughly 15 is typical.

Extra Solutions for Chapter 13

Solution to exercise 13.6 (p.216). With $\beta(f) = 2f^{1/2}(1-f)^{1/2}$, combining (13.14) and (13.25), the average probability of error of all linear codes is bounded by

$$\langle P(\text{block error})\rangle \leq \sum_{w>0} \langle A(w)\rangle [\beta(f)]^w \simeq \sum_{w>0} 2^{N[H_2(w/N)-(1-R)]} [\beta(f)]^w \tag{D.49}$$

This is a sum of terms that either grow or shrink exponentially with N, depending whether the first factor or the second dominates. We find the dominant term in the sum over w by differentiating the exponent.

$$\frac{\mathrm{d}}{\mathrm{d}w} N[H_2(w/N) - (1-R)] + w \log \beta(f) = \log \frac{1-(w/N)}{w/N} + \log \beta(f) \tag{D.50}$$

the maximum is at

$$\frac{w/N}{1-(w/N)} = \beta(f) \tag{D.51}$$

i.e.,

$$w/N = \frac{\beta(f)}{1+\beta(f)} = \frac{1}{1+1/\beta(f)}. \tag{D.52}$$

We require the exponent

$$N[H_2(w/N) - (1-R)] + w \log \beta(f) \tag{D.53}$$

to be negative at this point, then we can guarantee that the average error probability vanishes as N increases. Plugging in the maximum-achieving w/N, we have shown that the average error probability vanishes if

$$H_2\left(\frac{1}{1+1/\beta(f)}\right) + \frac{1}{1+1/\beta(f)} \log \beta(f) < (1-R), \tag{D.54}$$

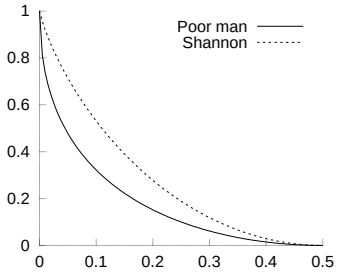


Figure D.5. Poor man's capacity (D.55) compared with Shannon's.

and we have thus proved a coding theorem, showing that reliable communication can be achieved over the binary symmetric channel at rates up to at least

$$R_{\text{poor man}} = 1 - \left[H_2 \left(\frac{1}{1 + 1/\beta(f)} \right) + \frac{1}{1 + 1/\beta(f)} \log \beta(f) \right]. \qquad \text{(D.55)}$$

Extra Solutions for Chapter 13

Solution to exercise 13.15 (p.221). All the Hamming codes have distance $d = 3$.

Solution to exercise 13.16 (p.221). A code has a word of weight 1 if an entire column of the parity-check matrix is zero. There is a chance of $2^{-M} = 2^{-360}$ that all entries in a given column are zero. There are $M = 360$ columns. So the expected value at $w = 1$ is

$$A(1) = M 2^{-M} = 360 \times 2^{-360} \simeq 10^{-111}. \qquad \text{(D.56)}$$

Solution to exercise 13.17 (p.221). This (15,5) code is unexpectedly good: While the Gilbert distance for a (15,5) code is 2.6, the minimum distance of the code is 7. The code can correct all errors of weight 1, 2, or 3. The weight enumerator function is (1,0,0,0,0,0,0,15,15,0,0,0,0,0,0,1).

Solution to exercise 13.18 (p.221). See figure D.6.

w	$A(w)$
0	1
5	12
6	10
8	15
9	20
10	6
Total	64

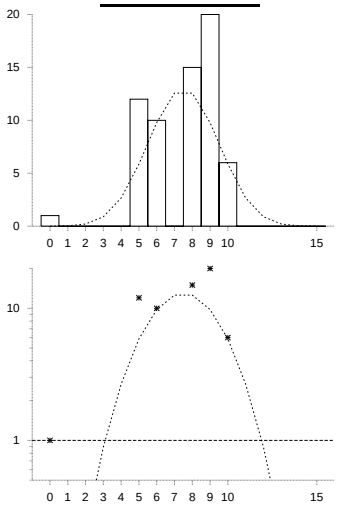


Figure D.6. The weight enumerator function of the pentagonful code (solid lines). The dotted lines show the average weight enumerator function of all random linear codes with the same size of generator matrix. The lower figure shows the same functions on a log scale. While the Gilbert distance is 2.2, the minimum distance of the code is 5.

Solution to exercise 13.25 (p.223). Here's a suggested attack on this still-open problem. [I use dual-containing as an abbreviation for "having a self-orthogonal dual".] Pick an ensemble of low-density parity-check codes – for example, defined by making an $M \times N$ matrix in which every column is a random vector of weight j. Each column involves $\binom{j}{2}$ pairs of rows. There are a total of $N\binom{j}{2}$ such pairs. If the code is dual-containing, every such pair must occur an even number of times, most probably twice.

Estimate the probability of every pair's occuring twice. Multiply this probability by the total number of codes in the ensemble to estimate the number that are dual-containing.

Solution to exercise 13.26 (p.223). The formula for the error probability produced by a single codeword of weight d is $\tilde{\Phi}(\sqrt{d}x)$, where x is the signal to noise ratio and $\tilde{\Phi}(u) = 1 - \Phi(u)$ is the tail area of a unit normal distribution. $E_{\text{b}}/N_0 = 10 \log_{10} \frac{x^2}{2R}$.

Extra Solutions for Chapter 15

Solution to exercise 15.1 (p.233).

(a) $\lceil \log_2 166751 \rceil = 18$ bits.

(b) 1.67×10^{-3}

Solution to exercise 15.2 (p.233).

(a) $H_2(0.4804) = 0.998891$.

(b) $0.496 \times H_2(0.5597) + 0.504 \times H_2(0.6) = 0.9802$ bits

(c) 1 bit.

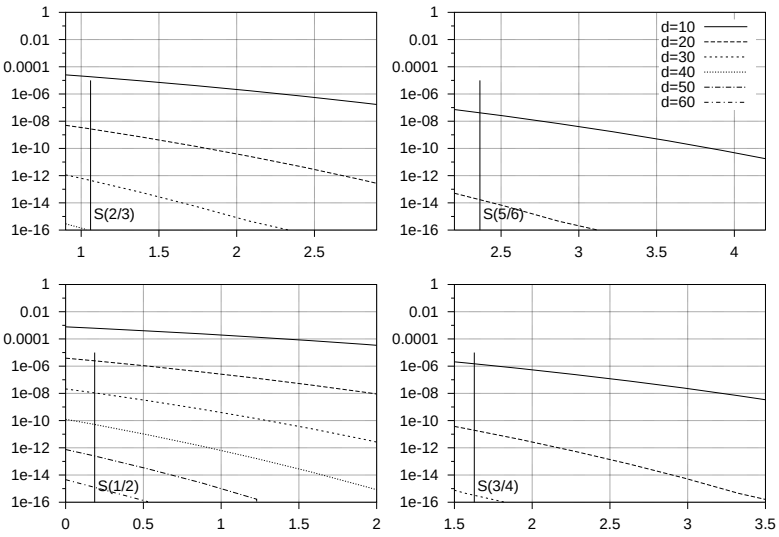


Figure D.7. Error probability associated with a single codeword of weight d as a function of the rate-compensated signal to noise ratio $E_{\rm b}/N_0$. Curves are shown for $d = 10, 20, \ldots$ and for $R = 1/2$, $2/3$, $3/4$, and $5/6$. In each plot the Shannon limit for a code of that rate is indicated by a vertical mark.

(d) $H_2(0.6) = 0.9709$ bits.

Solution to exercise 15.3 (p.233). The optimal symbol code (i.e., questioning strategy) has expected length $3\frac{11}{36}$.

Solution to exercise 15.8 (p.234). $|T| = 2716$.

Solution to exercise 15.4 (p.233). An arithmetic coding solution: use the coin to generate the bits of a binary real number between $0.000\ldots$ and $0.11111\ldots$; keep tossing until the number's position relative to $0.010101010\ldots$ and $0.101010101\ldots$ is apparent.

Interestingly, I think that the simple method

HH: Tom wins; HT: Dick wins; TH: Harry wins; TT: do over

is slightly more efficient in terms of the expected number of tosses.

Solution to exercise 15.11 (p.234). By symmetry, the optimal input distribution for the channel

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1-f & f \\ 0 & f & 1-f \end{bmatrix} \tag{D.57}$$

has the form $((1-p), p/2, p/2)$. The optimal input distribution is given by

$$p^* = \frac{1}{1 + 2^{H_2(f)-1}}. \tag{D.58}$$

In the case $f = 1/3$, $p^* = 0.514$ and the capacity is $C = 1.041$ bits.

Solution to exercise 15.15 (p.236). The optimal input distribution is $(1/6, 1/6, 1/3, 1/3)$, and the capacity is $\log_2 3$ bits.

More details for exercise 15.14 (p.235)

For the first part, for any x_{10}, $10x_{10} = (11-1)x_{10} = -x_{10} \bmod 11$, so $\text{sum}_1^9 = x_{10}$ implies $\text{sum}_1^9 + 10x_{10} = 0 \bmod 11$.

ISBN. Any change to a single digit violates the checksum.

Any interchange of two digits equal to a and b, separated by distance s in the word (for example, $s = 1$ for adjacent digits) produces a change in the checksum given by

$$[an + b(n+s) - (bn + a(n+s))] \bmod 11 = [bs - as] \bmod 11 = (b-a)s \bmod 11$$

Here s is between 1 and 9. And $b-a$ is between ± 9. If $b-a = 0$ then the digits are identical and their interchange doesn't matter. Now since 11 is prime, if $(b-a)s = 0 \bmod 11$, then $b - a = 0$. So all interchanges of two digits that matter can be detected.

If we used modulo 10 arithmetic then several things would go wrong. First, we would be unable to detect an interchange of the last two adjacent digits. For example 91 and 19 both check out, if they are the last two digits. Second, there would be other interchanges of pairs of digits which would be undetected because 10 is not prime. So for example, ...005... and ...500... would be indistinguishable. (This example uses two digits differing by 5 and separated by a space of size 2.) Third, a minor point: the probability of detecting a completely bogus ISBN is slightly higher (10/11) in the modulo 11 system than in the modulo 10 system (9/10).

More details for exercise ?? (p.??)

Let the transmitted string be $\mathbf{t}$ and the received string $\mathbf{r}$. The mutual information is:

$$H(\mathbf{t}; \mathbf{r}) = H(\mathbf{r}) - H(\mathbf{r}|\mathbf{t}). \tag{D.59}$$

Given the channel model, the conditional entropy $H(\mathbf{r}|\mathbf{t})$ is $\log_2(8) = 3$ bits, independent of the distribution chosen for $\mathbf{t}$.

By symmetry, the optimal input distribution is the uniform distribution, and this gives $H(\mathbf{r}) = 8$ bits.

So the capacity, which is the maximum mutual information, is [1]

$$C(Q) = 5 \,\text{bits}. \tag{D.60}$$

Encoder: A solution exists using a linear (8,5) code in which the first seven bits are constrained to be a codeword of the $(7, 4)$ Hamming code, which encodes 4 bits into 7 bits. The eighth transmitted bit is simply set to the fifth source bit.

Decoder: The decoder computes the syndrome of the first seven received bits using the 3×7 parity-check matrix of the Hamming code, and uses the normal Hamming code decoder to detect any single error in bits 1–7. If such an error is detected, the corresponding received bit is flipped, and the five source bits are read out. If on the other hand the syndrome is zero, then the final bit must be flipped.

Extra Solutions for Chapter 19

Theory of sex when the fitness is a sum of exclusive-ors

The following theory gives a reasonable fit to empirical data on evolution where the fitness function is a sum of exclusive-ors of independent pairs of

bits. Starting from random genomes, learning is initially slow because the populatin has to decide, for each pair of bits, in which direction to break the symmetry: should they go for `01` or `10`?

We approximate the situation by assuming that at time t, the fraction of the population that has `01` at a locus is $a(t)$, for all loci, and the fraction that have `10` is $d(t)$, for all loci. We thus assume that the symmetry gets broken in the same way at all loci. To ensure that this assumption loses no generality, we reserve the right to reorder the two bits. We assume that the other states at a locus, `00` and `11`, both appear in a fraction $b(t) \equiv \frac{1}{2}(1 - (a(t) + d(t)))$.

Now, we assume that all parents' genomes are drawn independently at random from this distribution. The probability distribution of the state of one locus in one child is then

$$\begin{aligned} P(\texttt{00}) &= b'(t) \equiv \frac{1}{2}(b + (a+b)(b+d)) \\ P(\texttt{01}) &= a'(t) \equiv \frac{1}{2}(a + (a+b)^2) \\ P(\texttt{10}) &= d'(t) \equiv \frac{1}{2}(d + (d+b)^2) \\ P(\texttt{11}) &= b'(t) \end{aligned} \tag{D.61}$$

where the first terms ($\frac{1}{2}b$, $\frac{1}{2}a$, etc.) come from the event that both bits inherited from a single parent.

The mean fitness of one locus in an offspring is then

$$p \equiv (a'(t) + d'(t)), \tag{D.62}$$

and the total fitness, which is the sum of $G/2$ such terms, has a binomial distribution with parameters $(N, p) = (G/2, p)$, i.e., mean $\mu = Np$ and variance $\sigma^2 = Np(1-p)$. Approximating this distribution by a Gaussian, and assuming truncation selection keeps the top half of the distribution, the mean fitness after truncation will be $\mu + \sqrt{2/\pi}\sigma$, and the fractions at one locus are adjusted, by this selection, to:

$$a''(t) \equiv a'(t)\frac{p''}{p}, \quad d''(t) \equiv d'(t)\frac{p''}{p} \tag{D.63}$$

where

$$p'' = p + \sqrt{2/\pi}\frac{1}{\sqrt{G/2}}\sqrt{p(1-p)}. \tag{D.64}$$

The parents of the next generation thus have fractions given by $a(t+1) = a''(t)$ and $d(t+1) = d''(t)$.

add graphs here from gene/xortheory

Extra Solutions for Chapter 22

Solution to exercise 22.15 (p.309). The likelihood has N maxima: it is infinitely large if μ is set equal to any datapoint x_n and σ_n is decreased to zero, the other $\sigma_{n'}$ being left at non-zero values. Notice also that the data's mean and median both give lousy answers to the question 'what is μ?'

We'll discuss the straightforward Bayesian solution to this problem later.

Extra Solutions for Chapter 29

Solution to exercise 29.1 (p.362). The solution in the book is incomplete, as the expression for the variance of

$$\hat{\Phi} \equiv \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}, \tag{D.65}$$

where

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})}, \tag{D.66}$$

is not given. We focus on the variance of the numerator. (The variance of the ratio is messier.)

But first, let's note the key insight here: what is the optimal $Q(x)$ going to look like? If $\phi(x)$ is a positive function, then the magic choice

$$Q(x) = \frac{1}{Z_Q} P^*(x)\phi(x) \tag{D.67}$$

(if we could make it) has the perfect property that every numerator term will evaulate to the same constant,

$$\frac{P^*(x^{(r)})}{Q^*(x^{(r)})}\phi(x^{(r)}) = \frac{P^*(x^{(r)})Z_Q}{P^*(x^{(r)})\phi(x^{(r)})}\phi(x^{(r)}) = Z_Q, \tag{D.68}$$

which is the required answer $Z_Q = \int \mathrm{d}x\, P^*(x)\phi(x)$. The choice (D.67) for Q thus minimizes the variance of the numerator. The denominators meanwhile would have the form

$$w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} = \frac{Z_Q}{\phi(x^{(r)})}. \tag{D.69}$$

It's intriguing to note that for this special choice of Q, where the numerator, even for just a single random point, is exactly the required answer, so that the best choice of denominator would be unity, the denominator created by the standard method is not unity (in general). This niggle exposes a general problem with importance sampling, which is that there are multiple possible expressions for the estimator, all of which are consistent asymptotically. Annoying, hey? The main motivation for estimators that include the denominator is so that the normalizing constants of the distributions P and Q do not need to be known.

So, to the variance. The variance of a single term in the numerator is, for normalized Q,

$$\mathrm{var}\left[\frac{P^*(x)}{Q(x)}\phi(x)\right] = \int \mathrm{d}x\, \left[\frac{P^*(x)}{Q(x)}\phi(x)\right]^2 Q(x) - \Phi^2 = \int \mathrm{d}x\, \frac{P^*(x)^2}{Q(x)}\phi(x)^2 - \Phi^2 \tag{D.70}$$

To minimize this variance with respect to Q, we can introduce a Lagrange multiplier λ to enforce normalization. The functional derivative with respect to $Q(x)$ is then

$$-\frac{P^*(x)^2}{Q(x)^2}\phi(x)^2 - \lambda, \tag{D.71}$$

which is zero if

$$Q(x) \propto P^*(x)|\phi(x)|. \tag{D.72}$$

Solution to exercise 29.14 (p.382). Fred's proposals would be appropriate if the target density $P(x)$ were half as great on the two end states as on all other states. If this were the target density, then the factor of two difference in Q for a transition in or out of an end state would be balanced by the factor of two difference in P, and the acceptance probability would be 1. Fred's algorithm therefore samples from the distribution

$$P'(x) = \begin{cases} 1/20 & x \in \{1, 2, \ldots, 19\} \\ 1/40 & x \in \{0, 20\} \\ 0 & \text{otherwise} \end{cases} . \tag{D.73}$$

If Fred wished to retain the new proposal density, he would have to change the acceptance rule such that transitions *out of* the end states would only be accepted with probability 0.5.

Solution to exercise 29.19 (p.384). Typical samples differ in their value of $\log P(\mathbf{x})$ by a standard deviation of order $\sqrt{N}$, let's say $c\sqrt{N}$. But the value of $\log P(\mathbf{x})$ varies during a Metropolis simulation by a random walk whose steps when negative are roughly of unit size; and thus by detailed balance the steps when positive are also roughly of unit size. So modelling the random walk of $\log P(\mathbf{x})$ as a drunkard's walk, it will take a time $T \simeq c^2 N$ to go a distance $c\sqrt{N}$ using unit steps.

Gibbs sampling will not necessarily take so long to generate independent samples because in Gibbs sampling it is possible for the value of $\log P(\mathbf{x})$ to change by a large quantity up or down in a single iteration. All the same, in many problems each Gibbs sampling update only changes $\log P(\mathbf{x})$ by a small amount of order 1, so $\log P(\mathbf{x})$ evolves by a random walk which takes a time $T \simeq c^2 N$ to traverse the typical set. However this linear scaling with the system size, N, is not unexpected – since Gibbs sampling updates only one coordinate at a time, we know that at least N updates (one for each variable) are needed to get to an independent point.

Extra Solutions for Chapter 31

Solution to exercise 31.3 (p.412). This is the problem of creating a system whose stable states are a desired set of memories. See later chapters for some ideas.

Extra Solutions for Chapter 35

Solution to exercise 35.3 (p.446). To answer this question, $P(x)$ can be transformed to a uniform density. Any property of intervals between record-breaking events that holds for the uniform density also holds for a general $P(x)$, since we can associate with any x a variable u equal to the cumulative probability density $\int^x P(x)$, and u's distribution is uniform. Whenever a record for x is broken, a record for u is broken also.

Solution to exercise 35.7 (p.448). Let's discuss the two possible parsings. The first parsing $\mathcal{H}_a$ produces a column of numbers all of which end in a decimal point. This might be viewed as a somewhat improbable parsing. Why is the decimal point there if no decimals follow it? On the other hand, this parsing makes every number four digits long, which has a pleasing and plausible simplicity to it.

However, if we use the second parsing $\mathcal{H}_b$ then the second column of numbers consists almost entirely of the number '0.0'. This also seems odd.

We could assign subjective priors to all these possibilities and suspicious coincidences. The most compelling evidence, however, comes from the fourth column of digits which are either the initial digits of a second list of numbers, or the final, post-decimal digits of the first list of numbers. What is the probability distribution of initial digits, and what is the probability distribution of final, post-decimal digits? It is often our experience that initial digits have a non-uniform distribution, with the digit '1' being much more probable than the digit '9'. Terminal digits often have a uniform distribution, or if they have a non-uniform distribution, it would be expected to be dominated either by '0' and '5' or by '0', '2', '4', '6', '8'. We don't generally expect the distribution of *terminal* digits to be asymmetric about '5', for example, we don't expect '2' and '8' to have very different probabilities.

The empirical distribution seems highly non-uniform and asymmetric, having 20 '1's, 21 '2's, one '3' and one '5'. This fits well with the hypothesis that the digits are initial digits (c.f. section 35.1), and does not fit well with any of the terminal digit distributions we thought of.

We can quantify the evidence in favour of the first hypothesis by picking a couple of crude assumptions: First, for initial digits,

$$P(n|\mathcal{H}_a) = \begin{cases} \frac{1}{Z}\frac{1}{n} & n \geq 1 \\ \frac{1}{Z}\frac{1}{10} & n = 0 \end{cases} , \tag{D.74}$$

where $Z = 2.93$, and second, for terminal digits,

$$P(n|\mathcal{H}_b) = \frac{1}{10}. \tag{D.75}$$

Then the probability of the given 43 digits is

$$P(\{n\}|\mathcal{H}_a) = 2.71 \times 10^{-28}. \tag{D.76}$$

$$P(\{n\}|\mathcal{H}_b) = 10^{-43}. \tag{D.77}$$

So the data consisting of the fourth column of digits favour $\mathcal{H}_a$ over $\mathcal{H}_b$ by about 10^{15} to 1.

This is an unreasonably extreme conclusion, as is typical of carelessly constructed Bayesian models (Mosteller and Wallace, 1984). But the conclusion is correct; the data are real data that I received from a colleague, and the correct parsing is that of $\mathcal{H}_a$.

Solution to exercise 35.8 (p.448). Bayes' theorem:

$$P(\mu|\{x_n\}) = \frac{P(\mu)\prod_n P(x_n|\mu)}{P(\{x_n\})} \tag{D.78}$$

The likelihood function contains a complete summary of what the experiment tells us about μ. The log likelihood,

$$L(\mu) = -\sum_n |x_n - \mu|, \tag{D.79}$$

is sketched in figure D.8.

The most probable values of μ are 0.9–2, and the posterior probability falls by a factor of e^2 once we reach -0.1 and 3, so a range of plausible 0 values for μ is $(-0.1, 3)$.

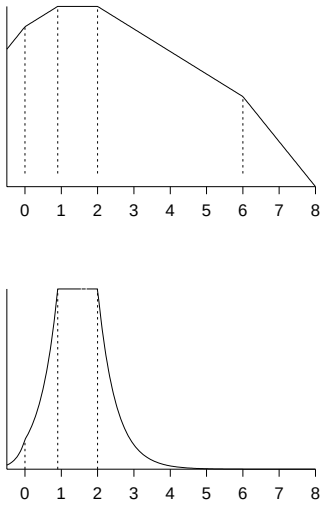


Figure D.8. Sketches of likelihood function. Top: likelihood function on a log scale. The gradient changes by 2 as we pass each data point. Gradients are 4, 2, 0, −2, −4. Bottom: likelihood function on a linear scale. The exponential functions have lengthscales 1/4, 1/2, 1/2, 1/4.

Extra Solutions for Chapter 36

Solution to exercise 36.5 (p.454).
Preference of A to B means

$$u(1) > .89u(1) + .10u(2.5) + .01u(0) \tag{D.80}$$

Whereas preference of D to C means

$$\begin{aligned} .89u(0) + .11u(1) &< .90u(0) + .10u(2.5) && \text{(D.81)} \\ .11u(1) &< .01u(0) + .10u(2.5) && \text{(D.82)} \\ u(1) &< .89u(1) + .10u(2.5) + .01u(0) && \text{(D.83)} \end{aligned}$$

which contradicts (D.80).

Solution to exercise 36.9 (p.456). The probability of winning either of the first two bets is $6/11 = 0.54545$. The probability that you win the third bet is 0.4544. Joe simply needs to make the third bet with a stake that is bigger than the sum of the first two stakes to have a positive expectation on the sequence of three bets.

The Las Vegas trickster

Solution to exercise 36.9 (p.456). On a single throw of the two dice, let the outcomes 6 and 7 have probabilities $P(6) = p_6$ and $P(7) = p_7$. Note $P(8) = p_6$. The values are $p_6 = 5/36$ and $p_7 = 6/36 = 1/6$. For the first bet, we can ignore other outcomes apart from the winning and losing outcomes 7 and 6 and compute the probability that the outcome is a 7, given that the game has terminated,

$$\frac{p_7}{p_6 + p_7} = 6/11 = 0.54545. \tag{D.84}$$

The second bet is identical. Both are favourable bets.

The third bet is the interesting one, because it is not a favourable bet for you, even though it sounds similar to the two bets that have gone before. The essential intuition for why two sevens are less probable than an 8 and a 6 is that the 8 and the 6 can come in either of two orders, so a rough factor of two appears in the probability for 8 and 6.

Computing the probability of winning is quite tricky if a neat route is not found. The probability is most easily computed if, as above, we *discard all the irrelevant events* and just compute the conditional probability of the different ways in which the state of the game can advance by one 'step'. The possible paths taken by this 'pruned' game with their probabilities are shown in the figure as a Markov process. (The original unpruned game is a similar Markov process in which an extra path emerges from each node, giving a transition back to the same node.) The node labelled 'A' denotes the initial state in which no 6s, 7s or 8s have been thrown. From here transitions are possible to state '7' in which exactly one 7 has been thrown, and no 6s or 8s; and to state 'E', in which either [one or more 8s have occurred and no 6s or 7s] or [one or more 6s have occurred and no 6s or 7s]. The probabilities of these transitions are shown. We can progress from state E only if Joe's winning 6 or 8 (whichever it is) is thrown, or if a 7 occurs. These events take us to the states labelled '68' and 'E7' respectively. From state '7' the game advances when a 6 or 8 is thrown, or when a 7 is thrown, taking us to states 'E7' and '77' respectively. Finally, from state E7, if a 7 is thrown we transfer to state

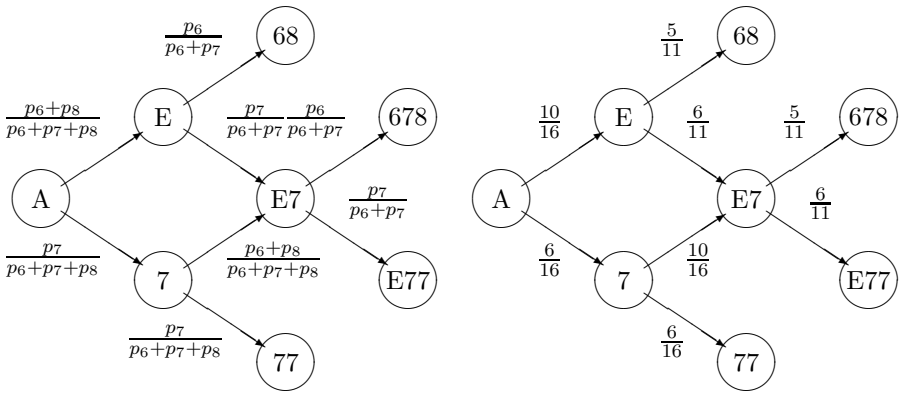


Figure D.9. Markov process describing the Las Vegas dice game, pruned of all irrelevant outcomes. The end states 68 and 678 are wins for Joe. States $E77$ and 77 are wins for you. Please do not confuse this state diagram, in which arrows indicate which states can follow from each other, with a graphical model, in which each node represents a different variables and arrows indicate causal relationships between them.

E77, and if Joe's required 6 or 8 is thrown, we move to state 678. States 68 and 678 are wins for Joe; states 77 and E77 are wins for you.

We first need the probability of state E7,

$$(10/16)(6/11) + (6/16)(10/16) = 405/704 = 0.5753 \tag{D.85}$$

The probability that you win is

$$P(77) + P(\mathrm{E77}) = (6/16)^2 + P(E7)(6/11) = 3519/7744 = 0.4544 \tag{D.86}$$

The bet is not favourable. Notice that Joe simply needs to make the third bet with a stake that is bigger than the sum of the first two stakes to have a positive expectation on the sequence of three bets.

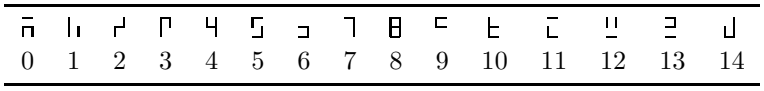
Extra Solutions for Chapter 39

Solution to exercise 39.1 (p.472). One answer, given in the text on page 472, is that the single neuron function was encountered under 'the best detection of pulses'. The same function has also appeared in the chapter on variational methods when we derived mean field theory for a spin system. Several of the solutions to the inference problems in chapter 1 were also written in terms of this function.

Solution to exercise 39.5 (p.480). If we let $\mathbf{x}$ and $\mathbf{s}$ be binary $\in \{\pm 1\}^7$, the likelihood is $(1-f)^N f^M$, where $N = (\mathbf{s}^{\mathsf{T}}\mathbf{x} + 7)/2$ and $M = (7 - \mathbf{s}^{\mathsf{T}}\mathbf{x})/2$. From here, it is straightforward to obtain the log posterior probability ratio, which is the activation.

The LED displays a binary code of length 7 with 10 codewords. Some codewords are very confusable – 8 and 9 differ by just one bit, for example. A superior binary code of length 7 is the $(7, 4)$ Hamming code. This code has 15 non-zero codewords, all separated by a distance of at least 3 bits.

Here are those 15 codewords, along with a suggested mapping to the integers 0–14.



Solution to exercise 39.6 (p.480).

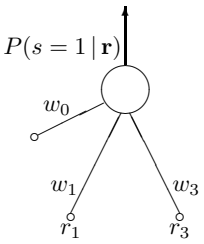
$$\begin{aligned}
\log \frac{P(s=1|\mathbf{r})}{P(s=2|\mathbf{r})} &= \log \frac{P(\mathbf{r}|s=1)P(s=1)}{P(\mathbf{r}|s=2)P(s=2)} \\
&= \log \left(\frac{1-f}{f}\right)^{2r_1-1} + \log \left(\frac{1-f}{f}\right)^{-(2r_3-1)} + \log \frac{P(s=1)}{P(s=2)} \\
&= w_1 r_1 + w_3 r_3 + w_0,
\end{aligned}$$

where

$$w_1 = 2\log\left(\frac{1-f}{f}\right), \quad w_3 = -2\log\left(\frac{1-f}{f}\right), \quad w_0 = \log\frac{P(s=1)}{P(s=2)}, \tag{D.87}$$

and $w_2 = 0$, which we can rearrange to give

$$P(s=1\,|\,\mathbf{r}) \;=\; \frac{1}{1+\exp\left(-w_0 - \sum_{n=1}^{3} w_n r_n\right)}.$$



This can be viewed as a neuron with two or three inputs, one from r_1 with a positive weight, and one from r_3 with a negative weight, and a bias.

Extra Solutions for Chapter 40

Solution to exercise 40.6 (p.490).

(a) $\mathbf{w} = (1, 1, 1)$.

(b) $\mathbf{w} = (1/4, 1/4, -1)$.

The two unrealizable labellings are $\{0, 0, 0, 1\}$ and $\{1, 1, 1, 0\}$.

Solution to exercise 40.8 (p.490). With just a little compression of the raw data, it's possible your brain could memorize everything.

Extra Solutions for Chapter 41

Solution to exercise 41.2 (p.502). When $\mathbf{w} \sim \text{Normal}(\mathbf{w}_{\text{MP}}, \mathbf{A}^{-1})$, the scalar $a = a(\mathbf{x}; \mathbf{w}_{\text{MP}}) + (\mathbf{w} - \mathbf{w}_{\text{MP}}) \cdot \mathbf{x}$ is Gaussian distributed with mean $a(\mathbf{x}; \mathbf{w}_{\text{MP}})$ and variance $s^2 = \mathbf{x}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{x}$.

This is easily shown by simply computing the mean and variance of a, then arguing that a's distribution must be Gaussian, because the marginals of a multivariate Gaussian are Gaussian. (See page 176 for a recap of multivariate Gaussians.) The mean is

$$\langle a \rangle = \langle a(\mathbf{x}; \mathbf{w}_{\text{MP}}) + (\mathbf{w} - \mathbf{w}_{\text{MP}}) \cdot \mathbf{x} \rangle = a(\mathbf{x}; \mathbf{w}_{\text{MP}}) + \langle (\mathbf{w} - \mathbf{w}_{\text{MP}}) \rangle \cdot \mathbf{x} = a(\mathbf{x}; \mathbf{w}_{\text{MP}}).$$

The variance is

$$\begin{aligned} \left\langle (a - a(\mathbf{x}; \mathbf{w}_{\text{MP}}))^2 \right\rangle &= \langle \mathbf{x} \cdot (\mathbf{w} - \mathbf{w}_{\text{MP}})(\mathbf{w} - \mathbf{w}_{\text{MP}}) \cdot \mathbf{x} \rangle \\ &= \mathbf{x}^{\mathsf{T}} \left\langle (\mathbf{w} - \mathbf{w}_{\text{MP}})(\mathbf{w} - \mathbf{w}_{\text{MP}})^{\mathsf{T}} \right\rangle \mathbf{x} \;=\; \mathbf{x}^{\mathsf{T}}\mathbf{A}^{-1}\mathbf{x}. \end{aligned} \tag{D.88}$$

Solution to exercise 41.3 (p.503). In the case of a single data point, the likelihood function, as a function of one parameter w_i, is a sigmoid function; an example of a sigmoid function is shown on a logarithmic scale in figure D.11a. The same figure shows a Gaussian distribution on a log scale. The prior distribution in this problem is assumed to be Gaussian; and the approximation Q is also a Gaussian, fitted at the maximum of the sum of the log likelihood and the log prior.

The log likelihood and log prior are both concave functions, so the curvature of $\log Q$ must necessarily be greater than the curvature of the log prior. But asymptotically the log likelihood function is linear, so the curvature of the log posterior for large $|a|$ decreases to the curvature of the log prior. Thus for sufficiently large values of w_i, the approximating distribution is *lighter-tailed* than the true posterior.

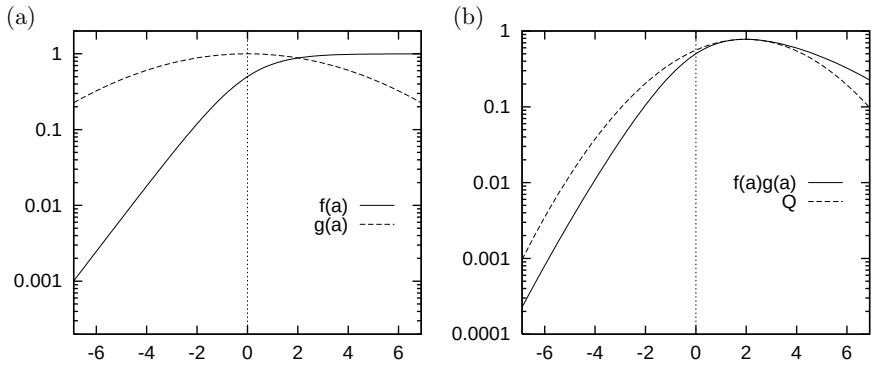


Figure D.11. (a) The log of the sigmoid function $f(a) = 1/(1 + e^{-a})$ and the log of a Gaussian $g(a) \propto \text{Normal}(0, 4^2)$. (b) The product $P = f(a)g(a)$ and a Gaussian approximation to it, fitted at its mode. Notice that for a range of negative values of a, the Gaussian approximation Q is bigger than P, while for values of a to the right of the mode, Q is smaller than P.

This conclusion may be a little misleading however. If we multiply the likelihood and the prior and find the maximum and fit a Gaussian there, we might obtain a picture like figure D.11b. Here issues of normalization have been ignored. The important point to note is that since the Gaussian is fitted at a point where the log likelihood's curvature is not very great, the approximating Gaussian's curvature is *too small* for a between a_{MP} and $-a_{\text{MP}}$, with the consequence that the approximation Q is substantially *larger* than P for a wide range of negative values of a. On the other hand, for values of a greater than a_{MP}, the approximation Q is smaller in value than P.

Thus whether Q is for practical purposes a heavy-tailed or light-tailed approximation to P depends which direction one looks in, and how far one looks.

The Gaussian approximation becomes most accurate when the amount of data increases, because the log of the posterior is a sum of more and more bent functions all of which contribute curvature to the log posterior, making it more and more Gaussian (c.f. figure 41.1). The greatest curvature is contributed by data points that are close (in terms of a) to the decision boundary, so the Gaussian approximation becomes good fastest if the optimized parameters are such that all the points are close to the decision boundary, that is, if the data are noisy.