About Chapter 13

In Chapters 8–11, we established Shannon's noisy-channel coding theorem for a general channel with any input and output alphabets. A great deal of attention in coding theory focuses on the special case of channels with binary inputs. Constraining ourselves to these channels simplifies matters, and leads us into an exceptionally rich world, which we will only taste in this book.

One of the aims of this chapter is to point out a contrast between Shannon's aim of achieving reliable communication over a noisy channel and the apparent aim of many in the world of coding theory. Many coding theorists take as their fundamental problem the task of packing as many spheres as possible, with radius as large as possible, into an N-dimensional space, *with no spheres overlapping.* Prizes are awarded to people who find packings that squeeze in an extra few spheres. While this is a fascinating mathematical topic, we shall see that the aim of maximizing the distance between codewords in a code has only a tenuous relationship to Shannon's aim of reliable communication.

13

Binary Codes

We've established Shannon's noisy-channel coding theorem for a general channel with any input and output alphabets. A great deal of attention in coding theory focuses on the special case of channels with binary inputs, the first implicit choice being the binary symmetric channel.

The optimal decoder for a code, given a binary symmetric channel, finds the codeword that is closest to the received vector, closest in *Hamming distance*. The Hamming distance between two binary vectors is the number of coordinates in which the two vectors differ. Decoding errors will occur if the noise takes us from the transmitted codeword $\mathbf{t}$ to a received vector $\mathbf{r}$ that is closer to some other codeword. The *distances* between codewords are thus relevant to the probability of a decoding error.

Example:
The Hamming distance
between `00001111`
and `11001101`
is 3.

▶ 13.1 Distance properties of a code

The *distance* of a code is the smallest separation between two of its codewords.

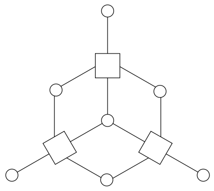
Example 13.1. The $(7,4)$ Hamming code (p.8) has distance $d = 3$. All pairs of its codewords differ in at least 3 bits. The maximum number of errors it can correct is $t = 1$; in general a code with distance d is $\lfloor (d-1)/2 \rfloor$-error-correcting.

A more precise term for distance is the *minimum distance* of the code. The distance of a code is often denoted by d or $d_{\min}$.

We'll now constrain our attention to linear codes. In a linear code, all codewords have identical distance properties, so we can summarize all the distances between the code's codewords by counting the distances from the all-zero codeword.

The *weight enumerator function* of a code, $A(w)$, is defined to be the number of codewords in the code that have weight w. The weight enumerator function is also known as the *distance distribution* of the code.

Example 13.2. The weight enumerator functions of the $(7,4)$ Hamming code and the dodecahedron code are shown in figures 13.1 and 13.2.



w	$A(w)$
0	1
3	7
4	7
7	1
Total	16

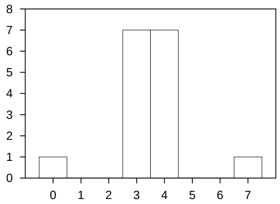
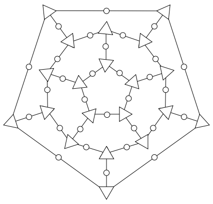


Figure 13.1. The graph of the $(7,4)$ Hamming code, and its weight enumerator function.

▶ 13.2 Obsession with distance

Since the maximum number of errors that a code can *guarantee* to correct, t, is related to its distance d by $t = \lfloor (d-1)/2 \rfloor$, many coding theorists focus on the distance of a code, searching for codes of a given size that have the biggest possible distance. Much of practical coding theory has focused on decoders that give the optimal decoding for all error patterns of weight up to the half-distance t of their codes.

$d = 2t + 1$ if d is odd, and
$d = 2t + 2$ if d is even.



w	$A(w)$
0	1
5	12
8	30
9	20
10	72
11	120
12	100
13	180
14	240
15	272
16	345
17	300
18	200
19	120
20	36
Total	2048

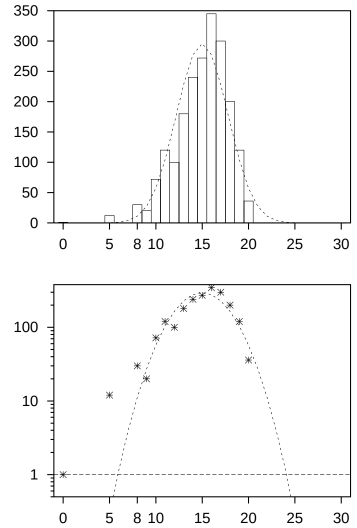


Figure 13.2. The graph defining the $(30, 11)$ dodecahedron code (the circles are the 30 transmitted bits and the triangles are the 20 parity checks, one of which is redundant) and the weight enumerator function (solid lines). The dotted lines show the average weight enumerator function of all random linear codes with the same size of generator matrix, which will be computed shortly. The lower figure shows the same functions on a log scale.

A bounded-distance decoder is a decoder that returns the closest codeword to a received binary vector $\mathbf{r}$ if the distance from $\mathbf{r}$ to that codeword is less than or equal to t; otherwise it returns a failure message.

The rationale for not trying to decode when more than t errors have occurred might be 'we can't *guarantee* that we can correct more than t errors, so we won't bother trying – who would be interested in a decoder that corrects some error patterns of weight greater than t, but not others?' This defeatist attitude is an example of *worst-case-ism*, a widespread mental ailment which this book is intended to cure.

The fact is that bounded-distance decoders cannot reach the Shannon limit of the binary symmetric channel; only a decoder that often corrects more than t errors can do this. The state of the art in error-correcting codes have decoders that work way beyond the minimum distance of the code.

$\ast$

Definitions of good and bad distance properties

Given a family of codes of increasing blocklength N, and with rates approaching a limit $R > 0$, we may be able to put that family in one of the following categories, which have some similarities to the categories of 'good' and 'bad' codes defined earlier (p.183):

A sequence of codes has 'good' distance if d/N tends to a constant greater than zero.

A sequence of codes has 'bad' distance if d/N tends to zero.

A sequence of codes has 'very bad' distance if d tends to a constant.

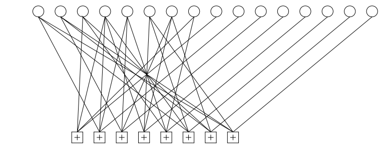


Figure 13.3. The graph of a rate-1/2 low-density generator-matrix code. The rightmost M of the transmitted bits are each connected to a single distinct parity constraint. The leftmost K transmitted bits are each connected to a small number of parity constraints.

Example 13.3. A *low-density generator-matrix code* is a linear code whose $K \times N$ generator matrix $\mathbf{G}$ has a small number d_0 of 1s per row, regardless of how big N is. The minimum distance of such a code is at most d_0, so low-density generator-matrix codes have 'very bad' distance.

While having large distance is no bad thing, we'll see, later on, why an emphasis on distance can be unhealthy.

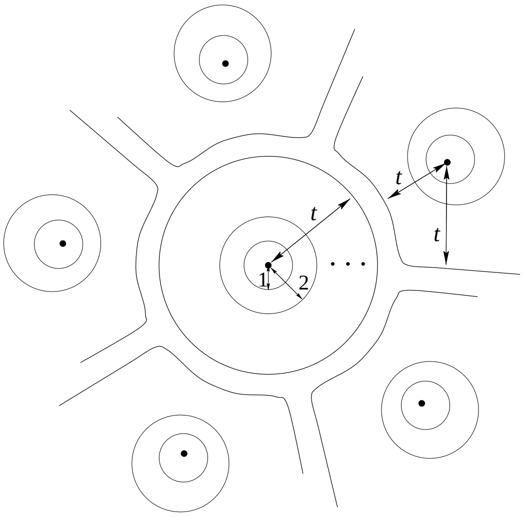


Figure 13.4. Schematic picture of part of Hamming space perfectly filled by t-spheres centred on the codewords of a perfect code.

▶ 13.3 Perfect codes

A t-sphere (or a sphere of radius t) in Hamming space, centred on a point $\mathbf{x}$, is the set of points whose Hamming distance from $\mathbf{x}$ is less than or equal to t.

The $(7, 4)$ Hamming code has the beautiful property that if we place 1-spheres about each of its 16 codewords, those spheres perfectly fill Hamming space without overlapping. As we saw in Chapter 1, every binary vector of length 7 is within a distance of $t = 1$ of exactly one codeword of the Hamming code.

A code is a perfect t-error-correcting code if the set of t-spheres centred on the codewords of the code fill the Hamming space without overlapping. (See figure 13.4.)

Let's recap our cast of characters. The number of codewords is $S = 2^K$. The number of points in the entire Hamming space is 2^N. The number of points in a Hamming sphere of radius t is

$$\sum_{w=0}^{t} \binom{N}{w}. \tag{13.1}$$

For a code to be perfect with these parameters, we require S times the number of points in the t-sphere to equal 2^N:

$$\text{for a perfect code,} \quad 2^K \sum_{w=0}^{t} \binom{N}{w} = 2^N \tag{13.2}$$

$$\text{or, equivalently,} \quad \sum_{w=0}^{t} \binom{N}{w} = 2^{N-K}. \tag{13.3}$$

For a perfect code, the number of noise vectors in one sphere must equal the number of possible syndromes. The $(7, 4)$ Hamming code satisfies this numerological condition because

$$1 + \binom{7}{1} = 2^3. \tag{13.4}$$

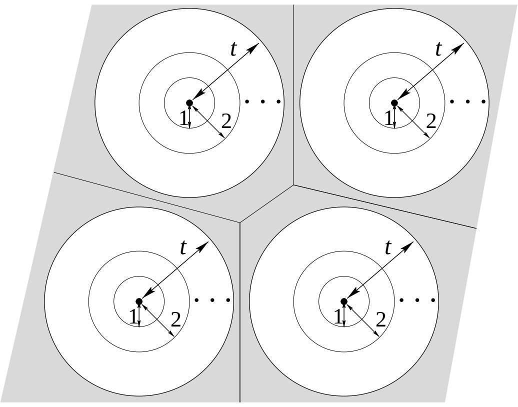


Figure 13.5. Schematic picture of Hamming space not perfectly filled by t-spheres centred on the codewords of a code. The grey regions show points that are at a Hamming distance of more than t from any codeword. This is a misleading picture, as, for any code with large t in high dimensions, the grey space between the spheres takes up almost all of Hamming space.

How happy we would be to use perfect codes

If there were large numbers of perfect codes to choose from, with a wide range of blocklengths and rates, then these would be the perfect solution to Shannon's problem. We could communicate over a binary symmetric channel with noise level f, for example, by picking a perfect t-error-correcting code with blocklength N and $t = f^*N$, where $f^* = f + \delta$ and N and δ are chosen such that the probability that the noise flips more than t bits is satisfactorily small.

However, *there are almost no perfect codes.* The only nontrivial perfect binary codes are ∗

1. the Hamming codes, which are perfect codes with $t = 1$ and blocklength $N = 2^M - 1$, defined below; the rate of a Hamming code approaches 1 as its blocklength N increases;

2. the repetition codes of odd blocklength N, which are perfect codes with $t = (N - 1)/2$; the rate of repetition codes goes to zero as $1/N$; and

3. one remarkable 3-error-correcting code with 2^{12} codewords of blocklength $N = 23$ known as the binary Golay code. [A second 2-error-correcting Golay code of length $N = 11$ over a ternary alphabet was discovered by a Finnish football-pool enthusiast called Juhani Virtakallio in 1947.]

There are no other binary perfect codes. Why this shortage of perfect codes? Is it because precise numerological coincidences like those satisfied by the parameters of the Hamming code (13.4) and the Golay code,

$$1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{11}, \tag{13.5}$$

are rare? Are there plenty of 'almost-perfect' codes for which the t-spheres fill *almost* the whole space?

No. In fact, the picture of Hamming spheres centred on the codewords *almost* filling Hamming space (figure 13.5) is a misleading one: for most codes, whether they are good codes or bad codes, almost all the Hamming space is taken up by the space *between* t-spheres (which is shown in grey in figure 13.5).

Having established this gloomy picture, we spend a moment filling in the properties of the perfect codes mentioned above.

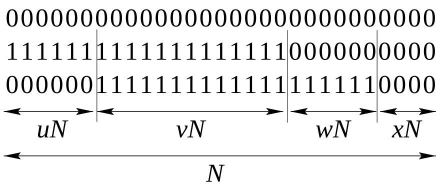


Figure 13.6. Three codewords.

The Hamming codes

The $(7,4)$ Hamming code can be defined as the linear code whose 3×7 parity-check matrix contains, as its columns, all the 7 $(= 2^3 - 1)$ non-zero vectors of length 3. Since these 7 vectors are all different, any single bit-flip produces a distinct syndrome, so all single-bit errors can be detected and corrected.

We can generalize this code, with $M = 3$ parity constraints, as follows. The Hamming codes are single-error-correcting codes defined by picking a number of parity-check constraints, M; the blocklength N is $N = 2^M - 1$; the parity-check matrix contains, as its columns, all the N non-zero vectors of length M bits.

The first few Hamming codes have the following rates:

Checks, M	(N,K)	$R = K/N$	
2	$(3,1)$	1/3	repetition code R_3
3	$(7,4)$	4/7	$(7,4)$ Hamming code
4	$(15,11)$	11/15	
5	$(31,26)$	26/31	
6	$(63,57)$	57/63	

Exercise 13.4.[2, p.223] What is the probability of block error of the (N,K) Hamming code to leading order, when the code is used for a binary symmetric channel with noise density f?

▶ 13.4 Perfectness is unattainable – first proof

We will show in several ways that useful perfect codes do not exist (here, 'useful' means 'having large blocklength N, and rate close neither to 0 nor 1').

Shannon proved that, given a binary symmetric channel with any noise level f, there exist codes with large blocklength N and rate as close as you like to $C(f) = 1 - H_2(f)$ that enable communication with arbitrarily small error probability. For large N, the number of errors per block will typically be about fN, so these codes of Shannon are 'almost-certainly-fN-error-correcting' codes.

Let's pick the special case of a noisy channel with $f \in (1/3, 1/2)$. Can we find a large *perfect* code that is fN-error-correcting? Well, let's suppose that such a code has been found, and examine just three of its codewords. (Remember that the code ought to have rate $R \simeq 1 - H_2(f)$, so it should have an enormous number (2^{NR}) of codewords.) Without loss of generality, we choose one of the codewords to be the all-zero codeword and define the other two to have overlaps with it as shown in figure 13.6. The second codeword differs from the first in a fraction $u + v$ of its coordinates. The third codeword differs from the first in a fraction $v + w$, and from the second in a fraction $u + w$. A fraction x of the coordinates have value zero in all three codewords. Now, if the code is fN-error-correcting, its minimum distance must be greater than $2fN$, so

$$u + v > 2f, \quad v + w > 2f, \quad \text{and} \quad u + w > 2f. \tag{13.6}$$

Summing these three inequalities and dividing by two, we have

$$u + v + w > 3f. \tag{13.7}$$

So if $f > 1/3$, we can deduce $u+v+w > 1$, so that $x < 0$, which is impossible. Such a code cannot exist. So the code cannot have *three* codewords, let alone 2^{NR}.

We conclude that, whereas Shannon proved there are plenty of codes for communicating over a binary symmetric channel with $f > 1/3$, *there are no perfect codes that can do this.*

We now study a more general argument that indicates that there are no large perfect linear codes for general rates (other than 0 and 1). We do this by finding the typical distance of a random linear code.

$$\left.\overbrace{\begin{bmatrix}
101010100100110100010110\\
001110111100011001101000\\
101110111001011000110100\\
000010111100101101001000\\
000000110011110100000100\\
110010001111100000101110\\
101111100010100001001110\\
110010110001101011101010\\
100011100101000010111101\\
010001000010101001101010\\
010111110111111110111010\\
101110101001001101000011
\end{bmatrix}}^{N}\right\} M$$

Figure 13.7. A random binary parity-check matrix.

▶ 13.5 Weight enumerator function of random linear codes

Imagine making a code by picking the binary entries in the $M \times N$ parity-check matrix $\mathbf{H}$ at random. What weight enumerator function should we expect?

The weight enumerator of one particular code with parity-check matrix $\mathbf{H}$, $A(w)_{\mathbf{H}}$, is the number of codewords of weight w, which can be written

$$A(w)_{\mathbf{H}} = \sum_{\mathbf{x}:|\mathbf{x}|=w} \mathbb{1}[\mathbf{H}\mathbf{x} = 0], \tag{13.8}$$

where the sum is over all vectors $\mathbf{x}$ whose weight is w and the truth function $\mathbb{1}[\mathbf{H}\mathbf{x} = 0]$ equals one if $\mathbf{H}\mathbf{x} = 0$ and zero otherwise.

We can find the expected value of $A(w)$,

$$\begin{aligned} \langle A(w) \rangle &= \sum_{\mathbf{H}} P(\mathbf{H}) A(w)_{\mathbf{H}} & (13.9)\\ &= \sum_{\mathbf{x}:|\mathbf{x}|=w} \sum_{\mathbf{H}} P(\mathbf{H})\, \mathbb{1}[\mathbf{H}\mathbf{x}=0], & (13.10)\end{aligned}$$

by evaluating the probability that a particular word of weight $w > 0$ is a codeword of the code (averaging over all binary linear codes in our ensemble). By symmetry, this probability depends only on the weight w of the word, not on the details of the word. The probability that the entire syndrome $\mathbf{H}\mathbf{x}$ is zero can be found by multiplying together the probabilities that each of the M bits in the syndrome is zero. Each bit z_m of the syndrome is a sum (mod 2) of w random bits, so the probability that $z_m = 0$ is $1/2$. The probability that $\mathbf{H}\mathbf{x} = 0$ is thus

$$\sum_{\mathbf{H}} P(\mathbf{H})\, \mathbb{1}[\mathbf{H}\mathbf{x}=0] = (1/2)^M = 2^{-M}, \tag{13.11}$$

independent of w.

The expected number of words of weight w (13.10) is given by summing, over all words of weight w, the probability that each word is a codeword. The number of words of weight w is $\binom{N}{w}$, so

$$\langle A(w) \rangle = \binom{N}{w} 2^{-M} \quad \text{for any } w > 0. \tag{13.12}$$

For large N, we can use $\log \binom{N}{w} \simeq NH_2(w/N)$ and $R \simeq 1 - M/N$ to write

$$\begin{aligned} \log_2 \langle A(w) \rangle &\simeq NH_2(w/N) - M & (13.13) \\ &\simeq N[H_2(w/N) - (1-R)] \text{ for any } w > 0. & (13.14) \end{aligned}$$

As a concrete example, figure 13.8 shows the expected weight enumerator function of a rate-1/3 random linear code with $N = 540$ and $M = 360$.

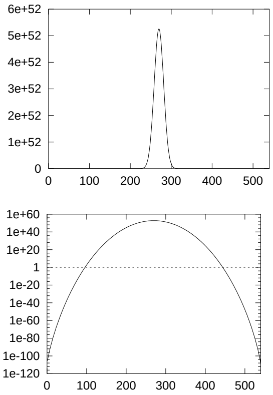


Figure 13.8. The expected weight enumerator function $\langle A(w) \rangle$ of a random linear code with $N = 540$ and $M = 360$. Lower figure shows $\langle A(w) \rangle$ on a logarithmic scale.

Gilbert–Varshamov distance

For weights w such that $H_2(w/N) < (1-R)$, the expectation of $A(w)$ is smaller than 1; for weights such that $H_2(w/N) > (1-R)$, the expectation is greater than 1. We thus expect, for large N, that the minimum distance of a random linear code will be close to the distance d_{GV} defined by

$$H_2(d_{\mathrm{GV}}/N) = (1-R). \tag{13.15}$$

Definition. This distance, $d_{\mathrm{GV}} \equiv NH_2^{-1}(1-R)$, is the *Gilbert–Varshamov distance* for rate R and blocklength N.

The *Gilbert–Varshamov conjecture*, widely believed, asserts that (for large N) it is not possible to create binary codes with minimum distance significantly greater than d_{GV}.

Definition. The *Gilbert–Varshamov rate* R_{GV} is the maximum rate at which you can reliably communicate with a bounded-distance decoder (as defined on p.207), assuming that the Gilbert–Varshamov conjecture is true.

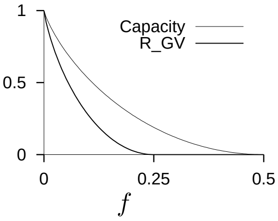


Figure 13.9. Contrast between Shannon's channel capacity C and the Gilbert rate R_{GV} – the maximum communication rate achievable using a bounded-distance decoder, as a function of noise level f. For any given rate, R, the maximum tolerable noise level for Shannon is twice as big as the maximum tolerable noise level for a 'worst-case-ist' who uses a bounded-distance decoder.

Why sphere-packing is a bad perspective, and an obsession with distance is inappropriate

If one uses a bounded-distance decoder, the maximum tolerable noise level will flip a fraction $f_{\mathrm{bd}} = \frac{1}{2} d_{\min}/N$ of the bits. So, assuming $d_{\min}$ is equal to the Gilbert distance d_{GV} (13.15), we have:

$$H_2(2f_{\mathrm{bd}}) = (1 - R_{\mathrm{GV}}). \tag{13.16}$$

$$R_{\mathrm{GV}} = 1 - H_2(2f_{\mathrm{bd}}). \tag{13.17}$$

Now, here's the crunch: what did Shannon say is achievable? He said the maximum possible rate of communication is the capacity,

$$C = 1 - H_2(f). \tag{13.18}$$

So for a given rate R, the maximum tolerable noise level, according to Shannon, is given by

$$H_2(f) = (1-R). \tag{13.19}$$

Our conclusion: imagine a good code of rate R has been chosen; equations (13.16) and (13.19) respectively define the maximum noise levels tolerable by a bounded-distance decoder, f_{bd}, and by Shannon's decoder, f.

$$f_{\mathrm{bd}} = f/2. \tag{13.20}$$

Bounded-distance decoders can only ever cope with *half* the noise-level that Shannon proved is tolerable!

How does this relate to perfect codes? A code is perfect if there are t-spheres around its codewords that fill Hamming space without overlapping.

But when a typical random linear code is used to communicate over a binary symmetric channel near to the Shannon limit, the typical number of bits flipped is fN, and the minimum distance between codewords is also fN, or a little bigger, if we are a little below the Shannon limit. So the fN-spheres around the codewords overlap with each other sufficiently that each sphere almost contains the centre of its nearest neighbour! The reason why this overlap is not disastrous is because, in high dimensions, the volume associated with the overlap, shown shaded in figure 13.10, is a tiny fraction of either sphere, so the probability of landing in it is extremely small.

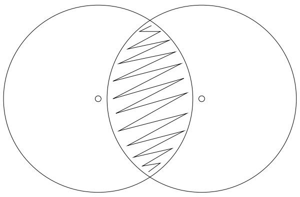


Figure 13.10. Two overlapping spheres whose radius is almost as big as the distance between their centres.

The moral of the story is that worst-case-ism can be bad for you, halving your ability to tolerate noise. You have to be able to decode *way* beyond the minimum distance of a code to get to the Shannon limit!

Nevertheless, the minimum distance of a code is of interest in practice, because, under some conditions, the minimum distance dominates the errors made by a code.

▶ 13.6 Berlekamp's bats

A blind bat lives in a cave. It flies about the centre of the cave, which corresponds to one codeword, with its typical distance from the centre controlled by a friskiness parameter f. (The displacement of the bat from the centre corresponds to the noise vector.) The boundaries of the cave are made up of stalactites that point in towards the centre of the cave (figure 13.11). Each stalactite is analogous to the boundary between the home codeword and another codeword. The stalactite is like the shaded region in figure 13.10, but reshaped to convey the idea that it is a region of very small volume.

Decoding errors correspond to the bat's intended trajectory passing inside a stalactite. Collisions with stalactites at various distances from the centre are possible.

If the friskiness is very small, the bat is usually very close to the centre of the cave; collisions will be rare, and when they do occur, they will usually involve the stalactites whose tips are closest to the centre point. Similarly, under low-noise conditions, decoding errors will be rare, and they will typically involve low-weight codewords. Under low-noise conditions, the minimum distance of a code is relevant to the (very small) probability of error.

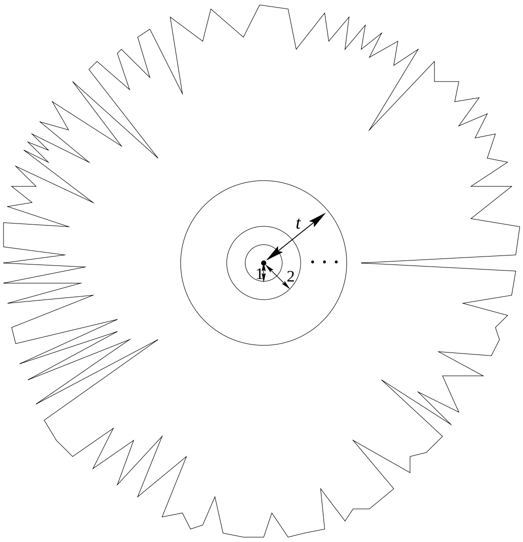


Figure 13.11. Berlekamp's schematic picture of Hamming space in the vicinity of a codeword. The jagged solid line encloses all points to which this codeword is the closest. The t-sphere around the codeword takes up a small fraction of this space.

If the friskiness is higher, the bat may often make excursions beyond the safe distance t where the longest stalactites start, but it will collide most frequently with more distant stalactites, owing to their greater number. There's only a tiny number of stalactites at the minimum distance, so they are relatively unlikely to cause the errors. Similarly, errors in a real error-correcting code depend on the properties of the weight enumerator function.

At very high friskiness, the bat is always a long way from the centre of the cave, and almost all its collisions involve contact with distant stalactites. Under these conditions, the bat's collision frequency has nothing to do with the distance from the centre to the closest stalactite.

▶ 13.7 Concatenation of Hamming codes

It is instructive to play some more with the concatenation of Hamming codes, a concept we first visited in figure 11.6, because we will get insights into the notion of good codes and the relevance or otherwise of the minimum distance of a code.

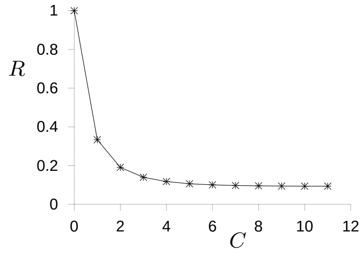


Figure 13.12. The rate R of the concatenated Hamming code as a function of the number of concatenations, C.

We can create a concatenated code for a binary symmetric channel with noise density f by encoding with several Hamming codes in succession.

The table recaps the key properties of the Hamming codes, indexed by number of constraints, M. All the Hamming codes have minimum distance $d = 3$ and can correct one error in N.

$N = 2^M - 1$	blocklength
$K = N - M$	number of source bits
$p_{\rm B} = \frac{3}{N}\binom{N}{2} f^2$	probability of block error to leading order

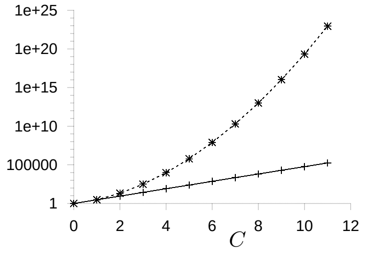


Figure 13.13. The blocklength N_C (upper curve) and minimum distance d_C (lower curve) of the concatenated Hamming code as a function of the number of concatenations C.

If we make a product code by concatenating a sequence of C Hamming codes with increasing M, we can choose those parameters $\{M_c\}_{c=1}^{C}$ in such a way that the rate of the product code

$$R_C = \prod_{c=1}^{C} \frac{N_c - M_c}{N_c} \tag{13.21}$$

tends to a non-zero limit as C increases. For example, if we set $M_1 = 2$, $M_2 = 3$, $M_3 = 4$, etc., then the asymptotic rate is 0.093 (figure 13.12).

The blocklength N is a rapidly-growing function of C, so these codes are somewhat impractical. A further weakness of these codes is that their minimum distance is not very good (figure 13.13). Every one of the constituent Hamming codes has minimum distance 3, so the minimum distance of the Cth product is 3^C. The blocklength N grows faster than 3^C, so the ratio d/N tends to zero as C increases. In contrast, for typical random codes, the ratio d/N tends to a constant such that $H_2(d/N) = 1 - R$. Concatenated Hamming codes thus have 'bad' distance.

Nevertheless, it turns out that this simple sequence of codes yields good codes for some channels – but not very good codes (see section 11.4 to recall the definitions of the terms 'good' and 'very good'). Rather than prove this result, we will simply explore it numerically.

Figure 13.14 shows the bit error probability $p_{\rm b}$ of the concatenated codes assuming that the constituent codes are decoded in sequence, as described in section 11.4. [This one-code-at-a-time decoding is suboptimal, as we saw there.] The horizontal axis shows the rates of the codes. As the number of concatenations increases, the rate drops to 0.093 and the error probability drops towards zero. The channel assumed in the figure is the binary symmetric channel with $f = 0.0588$. This is the highest noise level that can be tolerated using this concatenated code.

The take-home message from this story is *distance isn't everything.* The minimum distance of a code, although widely worshipped by coding theorists, is not of fundamental importance to Shannon's mission of achieving reliable communication over noisy channels.

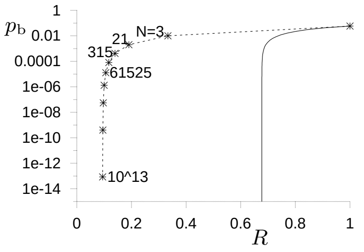


Figure 13.14. The bit error probabilities versus the rates R of the concatenated Hamming codes, for the binary symmetric channel with $f = 0.0588$. Labels alongside the points show the blocklengths, N. The solid line shows the Shannon limit for this channel. The bit error probability drops to zero while the rate tends to 0.093, so the concatenated Hamming codes are a 'good' code family.

▷ Exercise 13.5.[3] Prove that there exist families of codes with 'bad' distance that are 'very good' codes.

▶ 13.8 Distance isn't everything

Let's get a quantitative feeling for the effect of the minimum distance of a code, for the special case of a binary symmetric channel.

The error probability associated with one low-weight codeword

Let a binary code have blocklength N and just two codewords, which differ in d places. For simplicity, let's assume d is even. What is the error probability if this code is used on a binary symmetric channel with noise level f?

Bit flips matter only in places where the two codewords differ. The error probability is dominated by the probability that $d/2$ of these bits are flipped. What happens to the other bits is irrelevant, since the optimal decoder ignores them.

$$P(\text{block error}) \;\simeq\; \binom{d}{d/2} f^{d/2}(1-f)^{d/2}. \tag{13.22}$$

This error probability associated with a single codeword of weight d is plotted in figure 13.15. Using the approximation for the binomial coefficient (1.16), we can further approximate

$$\begin{aligned} P(\text{block error}) \;&\simeq\; \left[2f^{1/2}(1-f)^{1/2}\right]^d && (13.23)\\ &\equiv\; [\beta(f)]^d, && (13.24)\end{aligned}$$

where $\beta(f) = 2f^{1/2}(1-f)^{1/2}$ is called the Bhattacharyya parameter of the channel.

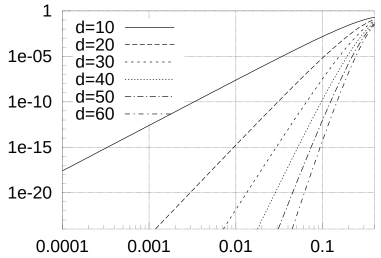


Figure 13.15. The error probability associated with a single codeword of weight d, $\binom{d}{d/2} f^{d/2}(1-f)^{d/2}$, as a function of f.

Now, consider a general linear code with distance d. Its block error probability must be at least $\binom{d}{d/2} f^{d/2}(1-f)^{d/2}$, independent of the blocklength N of the code. For this reason, a sequence of codes of increasing blocklength N and constant distance d (i.e., 'very bad' distance) cannot have a block error probability that tends to zero, on any binary symmetric channel. If we are interested in making superb error-correcting codes with tiny, tiny error probability, we might therefore shun codes with bad distance. However, being pragmatic, we should look more carefully at figure 13.15. In Chapter 1 we argued that codes for disk drives need an error probability smaller than about 10^{-18}. If the raw error probability in the disk drive is about 0.001, the error probability associated with one codeword at distance $d = 20$ is smaller than 10^{-24}. If the raw error probability in the disk drive is about 0.01, the error probability associated with one codeword at distance $d = 30$ is smaller than 10^{-20}. For practical purposes, therefore, it is not essential for a code to have good distance. For example, codes of blocklength 10 000, known to have many codewords of weight 32, can nevertheless correct errors of weight 320 with tiny error probability.

I wouldn't want you to think I am *recommending* the use of codes with bad distance; in Chapter 47 we will discuss low-density parity-check codes, my favourite codes, which have both excellent performance and *good* distance.

▶ 13.9 The union bound

The error probability of a code on the binary symmetric channel can be bounded in terms of its weight enumerator function by adding up appropriate multiples of the error probability associated with a single codeword (13.24):

$$P(\text{block error}) \leq \sum_{w>0} A(w)[\beta(f)]^w. \tag{13.25}$$

This inequality, which is an example of a union bound, is accurate for low noise levels f, but inaccurate for high noise levels, because it overcounts the contribution of errors that cause confusion with more than one codeword at a time.

▷ Exercise 13.6.[3] Poor man's noisy-channel coding theorem.

Pretending that the union bound (13.25) *is* accurate, and using the average weight enumerator function of a random linear code (13.14) (section 13.5) as $A(w)$, estimate the maximum rate $R_{\text{UB}}(f)$ at which one can communicate over a binary symmetric channel.

Or, to look at it more positively, using the union bound (13.25) as an inequality, show that communication at rates up to $R_{\text{UB}}(f)$ is possible over the binary symmetric channel.

In the following chapter, by analysing the probability of error of *syndrome decoding* for a binary linear code, and using a union bound, we will prove Shannon's noisy-channel coding theorem (for symmetric binary channels), and thus show that *very good linear codes exist.*

▶ 13.10 Dual codes

A concept that has some importance in coding theory, though we will have no immediate use for it in this book, is the idea of the dual of a linear error-correcting code.

An (N, K) linear error-correcting code can be thought of as a set of 2^K codewords generated by adding together all combinations of K independent basis codewords. The generator matrix of the code consists of those K basis codewords, conventionally written as row vectors. For example, the $(7, 4)$ Hamming code's generator matrix (from p.10) is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{13.26}$$

and its sixteen codewords were displayed in table 1.14 (p.9). The codewords of this code are linear combinations of the four vectors [1 0 0 0 1 0 1], [0 1 0 0 1 1 0], [0 0 1 0 1 1 1], and [0 0 0 1 0 1 1].

An (N, K) code may also be described in terms of an $M \times N$ parity-check matrix (where $M = N - K$) as the set of vectors $\{\mathbf{t}\}$ that satisfy

$$\mathbf{Ht} = \mathbf{0}. \tag{13.27}$$

One way of thinking of this equation is that each row of $\mathbf{H}$ specifies a vector to which $\mathbf{t}$ must be orthogonal if it is a codeword.

> The generator matrix specifies K vectors *from which* all codewords can be built, and the parity-check matrix specifies a set of M vectors *to which* all codewords are orthogonal.
>
> The dual of a code is obtained by exchanging the generator matrix and the parity-check matrix.

Definition. The set of *all* vectors of length N that are orthogonal to all codewords in a code, $\mathcal{C}$, is called the dual of the code, $\mathcal{C}^{\perp}$.

If $\mathbf{t}$ is orthogonal to $\mathbf{h}_1$ and $\mathbf{h}_2$, then it is also orthogonal to $\mathbf{h}_3 \equiv \mathbf{h}_1 + \mathbf{h}_2$; so all codewords are orthogonal to any linear combination of the M rows of $\mathbf{H}$. So the set of all linear combinations of the rows of the parity-check matrix is the dual code.

For our Hamming $(7,4)$ code, the parity-check matrix is (from p.12):

$$\mathbf{H} = \left[\begin{array}{cc} \mathbf{P} & \mathbf{I}_3 \end{array} \right] = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right]. \tag{13.28}$$

The dual of the $(7,4)$ Hamming code $\mathcal{H}_{(7,4)}$ is the code shown in table 13.16.

0000000	0101101	1001110	1100011
0010111	0111010	1011001	1110100

Table 13.16. The eight codewords of the dual of the $(7,4)$ Hamming code. [Compare with table 1.14, p.9.]

A possibly unexpected property of this pair of codes is that the dual, $\mathcal{H}^{\perp}_{(7,4)}$, is contained within the code $\mathcal{H}_{(7,4)}$ itself: every word in the dual code is a codeword of the original $(7,4)$ Hamming code. This relationship can be written using set notation:

$$\mathcal{H}^{\perp}_{(7,4)} \subset \mathcal{H}_{(7,4)}. \tag{13.29}$$

The possibility that the set of dual vectors can overlap the set of codeword vectors is counterintuitive if we think of the vectors as real vectors – how can a vector be orthogonal to itself? But when we work in modulo-two arithmetic, many non-zero vectors are indeed orthogonal to themselves!

▷ Exercise 13.7.[*1*, p.223] Give a simple rule that distinguishes whether a binary vector is orthogonal to itself, as is each of the three vectors [1 1 1 0 1 0 0], [0 1 1 1 0 1 0], and [1 0 1 1 0 0 1].

Some more duals

In general, if a code has a systematic generator matrix,

$$\mathbf{G} = [\mathbf{I}_K | \mathbf{P}^{\mathsf{T}}], \tag{13.30}$$

where $\mathbf{P}$ is a $K \times M$ matrix, then its parity-check matrix is

$$\mathbf{H} = [\mathbf{P} | \mathbf{I}_M]. \tag{13.31}$$

Example 13.8. The repetition code R_3 has generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}; \tag{13.32}$$

its parity-check matrix is

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{13.33}$$

The two codewords are [1 1 1] and [0 0 0].

The dual code has generator matrix

$$\mathbf{G}^{\perp} = \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{13.34}$$

or equivalently, modifying $\mathbf{G}^{\perp}$ into systematic form by row additions,

$$\mathbf{G}^{\perp} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{13.35}$$

We call this dual code the *simple parity code* P_3; it is the code with one parity-check bit, which is equal to the sum of the two source bits. The dual code's four codewords are [1 1 0], [1 0 1], [0 0 0], and [0 1 1].

In this case, the only vector common to the code and the dual is the all-zero codeword.

Goodness of duals

If a sequence of codes is 'good', are their duals good too? Examples can be constructed of all cases: good codes with good duals (random linear codes); bad codes with bad duals; and good codes with bad duals. The last category is especially important: many state-of-the-art codes have the property that their duals are bad. The classic example is the low-density parity-check code, whose dual is a low-density generator-matrix code.

▷ Exercise 13.9.[3] Show that low-density generator-matrix codes are bad. A family of low-density generator-matrix codes is defined by two parameters j, k, which are the column weight and row weight of all rows and columns respectively of $\mathbf{G}$. These weights are fixed, independent of N; for example, $(j, k) = (3, 6)$. [Hint: show that the code has low-weight codewords, then use the argument from p.215.]

Exercise 13.10.[5] Show that low-density parity-check codes are good, and have good distance. (For solutions, see Gallager (1963) and MacKay (1999b).)

Self-dual codes

The $(7, 4)$ Hamming code had the property that the dual was contained in the code itself. A code is *self-orthogonal* if it is contained in its dual. For example, the dual of the $(7, 4)$ Hamming code is a self-orthogonal code. One way of seeing this is that the overlap between any pair of rows of $\mathbf{H}$ is even. Codes that contain their duals are important in quantum error-correction (Calderbank and Shor, 1996).

It is intriguing, though not necessarily useful, to look at codes that are *self-dual*. A code $\mathcal{C}$ is self-dual if the dual of the code is identical to the code.

$$\mathcal{C}^{\perp} = \mathcal{C}. \tag{13.36}$$

Some properties of self-dual codes can be deduced:

1. If a code is self-dual, then its generator matrix is also a parity-check matrix for the code.
2. Self-dual codes have rate 1/2, i.e., $M = K = N/2$.
3. All codewords have even weight.

▷ Exercise 13.11.$^{[2, \text{ p.223}]}$ What property must the matrix $\mathbf{P}$ satisfy, if the code with generator matrix $\mathbf{G} = [\mathbf{I}_K|\mathbf{P}^{\mathsf{T}}]$ is self-dual?

Examples of self-dual codes

1. The repetition code R_2 is a simple example of a self-dual code.

$$\mathbf{G} = \mathbf{H} = \left[\begin{array}{cc} 1 & 1 \end{array}\right]. \tag{13.37}$$

2. The smallest non-trivial self-dual code is the following $(8,4)$ code.

$$\mathbf{G} = \left[\ \mathbf{I}_4 \ \middle|\ \mathbf{P}^{\mathsf{T}}\ \right] = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right]. \tag{13.38}$$

▷ Exercise 13.12.$^{[2, \text{ p.223}]}$ Find the relationship of the above $(8,4)$ code to the $(7,4)$ Hamming code.

Duals and graphs

Let a code be represented by a graph in which there are nodes of two types, parity-check constraints and equality constraints, joined by edges which represent the bits of the code (not all of which need be transmitted).

The dual code's graph is obtained by replacing all parity-check nodes by equality nodes and *vice versa*. This type of graph is called a normal graph by Forney (2001).

Further reading

Duals are important in coding theory because functions involving a code (such as the posterior distribution over codewords) can be transformed by a Fourier transform into functions over the dual code. For an accessible introduction to Fourier analysis on finite groups, see Terras (1999). See also MacWilliams and Sloane (1977).

▶ 13.11 Generalizing perfectness to other channels

Having given up on the search for perfect codes for the binary symmetric channel, we could console ourselves by changing channel. We could call a code 'a perfect u-error-correcting code for the binary erasure channel' if it can restore any u erased bits, and never more than u. Rather than using the word perfect, however, the conventional term for such a code is a 'maximum distance separable code', or MDS code.

In a perfect u-error-correcting code for the binary erasure channel, the number of redundant bits must be $N - K = u$.

As we already noted in exercise 11.10 (p.190), the $(7,4)$ Hamming code is *not* an MDS code. It can recover *some* sets of 3 erased bits, but not all. If any 3 bits corresponding to a codeword of weight 3 are erased, then one bit of information is unrecoverable. This is why the $(7,4)$ code is a poor choice for a RAID system.

A tiny example of a maximum distance separable code is the simple parity-check code P_3 whose parity-check matrix is $\mathbf{H} = [1\,1\,1]$. This code has 4 codewords, all of which have even parity. All codewords are separated by a distance of 2. Any single erased bit can be restored by setting it to the parity of the other two bits. The repetition codes are also maximum distance separable codes.

▷ Exercise 13.13.[*5*, p.224] Can you make an (N, K) code, with $M = N - K$ parity symbols, for a q-ary erasure channel, such that the decoder can recover the codeword when *any* M symbols are erased in a block of N? [Example: for the channel with $q = 4$ symbols there is an $(N, K) = (5, 2)$ code which can correct any $M = 3$ erasures.]

For the q-ary erasure channel with $q > 2$, there are large numbers of MDS codes, of which the Reed–Solomon codes are the most famous and most widely used. As long as the field size q is bigger than the blocklength N, MDS block codes of any rate can be found. (For further reading, see Lin and Costello (1983).)

▶ 13.12 Summary

Shannon's codes for the binary symmetric channel can almost always correct fN errors, but they are not fN-error-correcting codes.

Reasons why the distance of a code has little relevance

1. The Shannon limit shows that the best codes must be able to cope with a noise level twice as big as the maximum noise level for a bounded-distance decoder.

2. When the binary symmetric channel has $f > 1/4$, no code with a bounded-distance decoder can communicate at all; but Shannon says good codes exist for such channels.

3. Concatenation shows that we can get good performance even if the distance is bad.

The whole weight enumerator function is relevant to the question of whether a code is a good code.

The relationship between good codes and distance properties is discussed further in exercise 13.14 (p.220).

▶ 13.13 Further exercises

Exercise 13.14.[*3*, p.224] A codeword $\mathbf{t}$ is selected from a linear (N, K) code $\mathcal{C}$, and it is transmitted over a noisy channel; the received signal is $\mathbf{y}$. We assume that the channel is a memoryless channel such as a Gaussian channel. Given an assumed channel model $P(\mathbf{y} \mid \mathbf{t})$, there are two decoding problems.

The codeword decoding problem is the task of inferring which codeword $\mathbf{t}$ was transmitted given the received signal.

The bitwise decoding problem is the task of inferring for each transmitted bit t_n how likely it is that that bit was a one rather than a zero.

Consider optimal decoders for these two decoding problems. Prove that the probability of error of the optimal bitwise-decoder is closely related to the probability of error of the optimal codeword-decoder, by proving the following theorem.

Theorem 13.1 *If a binary linear code has minimum distance $d_{\rm min}$, then, for any given channel, the codeword bit error probability of the optimal bitwise decoder, $p_{\rm b}$, and the block error probability of the maximum likelihood decoder, $p_{\rm B}$, are related by:*

$$p_{\rm B} \geq p_{\rm b} \geq \frac{1}{2}\frac{d_{\rm min}}{N} p_{\rm B}. \tag{13.39}$$

Exercise 13.15.$^{[1]}$ What are the minimum distances of the $(15, 11)$ Hamming code and the $(31, 26)$ Hamming code?

▷ Exercise 13.16.$^{[2]}$ Let $A(w)$ be the average weight enumerator function of a rate-1/3 random linear code with $N = 540$ and $M = 360$. Estimate, from first principles, the value of $A(w)$ at $w = 1$.

Exercise 13.17.$^{[3C]}$ A code with minimum distance greater than $d_{\rm GV}$. A rather nice $(15, 5)$ code is generated by this generator matrix, which is based on measuring the parities of all the $\binom{5}{3} = 10$ triplets of source bits:

$$\mathbf{G} = \left[\begin{array}{ccccccccccccccc}
1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot & 1 & 1 & \cdot & 1 \\
\cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 & \cdot & 1 & 1 & \cdot \\
\cdot & \cdot & 1 & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot & 1 & \cdot & 1 & 1 \\
\cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & \cdot & 1 & 1 & \cdot & 1 & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & 1 & 1 & 1 & 1 & \cdot & \cdot & 1 & 1 & \cdot & 1 & \cdot
\end{array}\right]. \tag{13.40}$$

Find the minimum distance and weight enumerator function of this code.

Exercise 13.18.$^{[3C]}$ Find the minimum distance of the 'pentagonful' low-density parity-check code whose parity-check matrix is

$$\mathbf{H} = \left[\begin{array}{ccccc|ccccc|ccccc}
1 & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1
\end{array}\right]. \tag{13.41}$$

Show that nine of the ten rows are independent, so the code has parameters $N = 15$, $K = 6$. Using a computer, find its weight enumerator function.

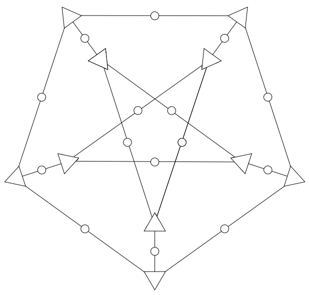


Figure 13.17. The graph of the pentagonful low-density parity-check code with 15 bit nodes (circles) and 10 parity-check nodes (triangles). [This graph is known as the Petersen graph.]

▷ Exercise 13.19.$^{[3C]}$ Replicate the calculations used to produce figure 13.12. Check the assertion that the highest noise level that's correctable is 0.0588. Explore alternative concatenated sequences of codes. Can you find a better sequence of concatenated codes – better in the sense that it has either higher asymptotic rate R or can tolerate a higher noise level f?

Exercise 13.20.[3, p.226] Investigate the possibility of achieving the Shannon limit with linear block codes, using the following counting argument. Assume a linear code of large blocklength N and rate $R = K/N$. The code's parity-check matrix $\mathbf{H}$ has $M = N - K$ rows. Assume that the code's optimal decoder, which solves the syndrome decoding problem $\mathbf{Hn} = \mathbf{z}$, allows reliable communication over a binary symmetric channel with flip probability f.

How many 'typical' noise vectors $\mathbf{n}$ are there?

Roughly how many distinct syndromes $\mathbf{z}$ are there?

Since $\mathbf{n}$ is reliably deduced from $\mathbf{z}$ by the optimal decoder, the number of syndromes must be greater than or equal to the number of typical noise vectors. What does this tell you about the largest possible value of rate R for a given f?

▷ Exercise 13.21.[2] Linear binary codes use the input symbols `0` and `1` with equal probability, implicitly treating the channel as a symmetric channel. Investigate how much loss in communication rate is caused by this assumption, if in fact the channel is a highly asymmetric channel. Take as an example a Z-channel. How much smaller is the maximum possible rate of communication using symmetric inputs than the capacity of the channel? [Answer: about 6%.]

Exercise 13.22.[2] Show that codes with 'very bad' distance are 'bad' codes, as defined in section 11.4 (p.183).

Exercise 13.23.[3] One linear code can be obtained from another by *puncturing*. Puncturing means taking each codeword and deleting a defined set of bits. Puncturing turns an (N, K) code into an (N', K) code, where $N' < N$.

Another way to make new linear codes from old is *shortening*. Shortening means constraining a defined set of bits to be zero, and then deleting them from the codewords. Typically if we shorten by one bit, half of the code's codewords are lost. Shortening typically turns an (N, K) code into an (N', K') code, where $N - N' = K - K'$.

Another way to make a new linear code from two old ones is to make the *intersection* of the two codes: a codeword is only retained in the new code if it is present in both of the two old codes.

Discuss the effect on a code's distance-properties of puncturing, shortening, and intersection. Is it possible to turn a code family with bad distance into a code family with good distance, or *vice versa*, by each of these three manipulations?

Exercise 13.24.[3, p.226] Todd Ebert's 'hat puzzle'.

Three players enter a room and a red or blue hat is placed on each person's head. The colour of each hat is determined by a coin toss, with the outcome of one coin toss having no effect on the others. Each person can see the other players' hats but not his own.

No communication of any sort is allowed, except for an initial strategy session before the group enters the room. Once they have had a chance to look at the other hats, the players must simultaneously guess their own hat's colour or pass. The group shares a \$3 million prize if *at least one player guesses correctly and no players guess incorrectly.*

The same game can be played with any number of players. The general problem is to find a strategy for the group that maximizes its chances of winning the prize. Find the best strategies for groups of size **three** and **seven**.

[Hint: when you've done **three** and **seven**, you might be able to solve **fifteen**.]

If you already know the hat puzzle, you could try the 'Scottish version' of the rules in which the prize is only awarded to the group if they *all* guess correctly.

In the 'Reformed Scottish version', all the players must guess correctly, and there are *two* rounds of guessing. Those players who guess during round one leave the room. The remaining players must guess in round two. What strategy should the team adopt to maximize their chance of winning?

Exercise 13.25.$^{[5]}$ Estimate how many binary low-density parity-check codes have self-orthogonal duals. [Note that we don't expect a huge number, since almost all low-density parity-check codes are 'good', but a low-density parity-check code that contains its dual must be 'bad'.]

Exercise 13.26.$^{[2C]}$ In figure 13.15 we plotted the error probability associated with a single codeword of weight d as a function of the noise level f of a binary symmetric channel. Make an equivalent plot for the case of the Gaussian channel, showing the error probability associated with a single codeword of weight d as a function of the rate-compensated signal-to-noise ratio $E_{\rm b}/N_0$. Because $E_{\rm b}/N_0$ depends on the rate, you have to choose a code rate. Choose $R = 1/2$, $2/3$, $3/4$, or $5/6$.

▶ 13.14 Solutions

Solution to exercise 13.4 (p.210). The probability of block error to leading order is $p_{\rm B} = \frac{3}{N}\binom{N}{2} f^2$.

Solution to exercise 13.7 (p.217). A binary vector is perpendicular to itself if it has even weight, i.e., an even number of 1s.

Solution to exercise 13.11 (p.219). The self-dual code has two equivalent parity-check matrices, $\mathbf{H}_1 = \mathbf{G} = [\mathbf{I}_K | \mathbf{P}^{\sf T}]$ and $\mathbf{H}_2 = [\mathbf{P} | \mathbf{I}_K]$; these must be equivalent to each other through row additions, that is, there is a matrix $\mathbf{U}$ such that $\mathbf{U}\mathbf{H}_2 = \mathbf{H}_1$, so

$$[\mathbf{U}\mathbf{P} | \mathbf{U}\mathbf{I}_K] = [\mathbf{I}_K | \mathbf{P}^{\sf T}] \,. \tag{13.42}$$

From the right-hand sides of this equation, we have $\mathbf{U} = \mathbf{P}^{\sf T}$, so the left-hand sides become:

$$\mathbf{P}^{\sf T}\mathbf{P} = \mathbf{I}_K. \tag{13.43}$$

Thus if a code with generator matrix $\mathbf{G} = [\mathbf{I}_K | \mathbf{P}^{\sf T}]$ is self-dual then $\mathbf{P}$ is an *orthogonal* matrix, modulo 2, and *vice versa.*

Solution to exercise 13.12 (p.219). The $(8, 4)$ and $(7, 4)$ codes are intimately related. The $(8, 4)$ code, whose parity-check matrix is

$$\mathbf{H} = \left[\; \mathbf{P} \;\middle|\; \mathbf{I}_4 \;\right] = \left[\begin{array}{cccc|cccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right], \tag{13.44}$$

is obtained by (a) appending an extra parity-check bit which can be thought of as the parity of all seven bits of the $(7, 4)$ Hamming code; and (b) reordering the first four bits.

Solution to exercise 13.13 (p.220). If an (N, K) code, with $M = N - K$ parity symbols, has the property that the decoder can recover the codeword when *any* M symbols are erased in a block of N, then the code is said to be maximum distance separable (MDS).

No MDS binary codes exist, apart from the repetition codes and simple parity codes. For $q > 2$, some MDS codes can be found.

As a simple example, here is a $(9, 2)$ code for the 8-ary erasure channel. The code is defined in terms of the multiplication and addition rules of $GF(8)$, which are given in Appendix C.1. The elements of the input alphabet are $\{0, 1, A, B, C, D, E, F\}$ and the generator matrix of the code is

$$\mathbf{G} = \left[\begin{array}{ccccccccc} 1 & 0 & 1 & A & B & C & D & E & F \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right]. \tag{13.45}$$

The resulting 64 codewords are:

```
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
```

Solution to exercise 13.14 (p.220). Quick, rough proof of the theorem. Let $\mathbf{x}$ denote the difference between the reconstructed codeword and the transmitted codeword. For any given channel output $\mathbf{r}$, there is a posterior distribution over $\mathbf{x}$. This posterior distribution is positive only on vectors $\mathbf{x}$ belonging to the code; the sums that follow are over codewords $\mathbf{x}$. The block error probability is:

$$p_{\mathrm{B}} = \sum_{\mathbf{x} \neq 0} P(\mathbf{x} \,|\, \mathbf{r}). \tag{13.46}$$

The average bit error probability, averaging over all bits in the codeword, is:

$$p_{\mathrm{b}} = \sum_{\mathbf{x} \neq 0} P(\mathbf{x} \,|\, \mathbf{r}) \frac{w(\mathbf{x})}{N}, \tag{13.47}$$

where $w(\mathbf{x})$ is the weight of codeword $\mathbf{x}$. Now the weights of the non-zero codewords satisfy

$$1 \geq \frac{w(\mathbf{x})}{N} \geq \frac{d_{\mathrm{min}}}{N}. \tag{13.48}$$

Substituting the inequalities (13.48) into the definitions (13.46, 13.47), we obtain:

$$p_{\mathrm{B}} \geq p_{\mathrm{b}} \geq \frac{d_{\mathrm{min}}}{N} p_{\mathrm{B}}, \tag{13.49}$$

which is a factor of two stronger, on the right, than the stated result (13.39). In making the proof watertight, I have weakened the result a little.

Careful proof. The theorem relates the performance of the optimal block decoding algorithm and the optimal bitwise decoding algorithm.

We introduce another pair of decoding algorithms, called the block-guessing decoder and the bit-guessing decoder. The idea is that these two algorithms are similar to the optimal block decoder and the optimal bitwise decoder, but lend themselves more easily to analysis.

We now define these decoders. Let $\mathbf{x}$ denote the inferred codeword. For any given code:

The optimal block decoder returns the codeword $\mathbf{x}$ that maximizes the posterior probability $P(\mathbf{x} \,|\, \mathbf{r})$, which is proportional to the likelihood $P(\mathbf{r} \,|\, \mathbf{x})$.

The probability of error of this decoder is called $p_{\rm B}$.

The optimal bit decoder returns for each of the N bits, x_n, the value of a that maximizes the posterior probability $P(x_n = a \,|\, \mathbf{r}) = \sum_{\mathbf{x}} P(\mathbf{x} \,|\, \mathbf{r}) \, \mathbb{1}[x_n = a]$.

The probability of error of this decoder is called $p_{\rm b}$.

The block-guessing decoder returns a random codeword $\mathbf{x}$ with probability distribution given by the posterior probability $P(\mathbf{x} \,|\, \mathbf{r})$.

The probability of error of this decoder is called $p_{\rm B}^{\rm G}$.

The bit-guessing decoder returns for each of the N bits, x_n, a random bit from the probability distribution $P(x_n = a \,|\, \mathbf{r})$.

The probability of error of this decoder is called $p_{\rm b}^{\rm G}$.

The theorem states that the optimal bit error probability $p_{\rm b}$ is bounded above by $p_{\rm B}$ and below by a given multiple of $p_{\rm B}$ (13.39).

The left-hand inequality in (13.39) is trivially true – if a block is correct, all its constituent bits are correct; so if the optimal block decoder outperformed the optimal bit decoder, we could make a better bit decoder from the block decoder.

We prove the right-hand inequality by establishing that:

(a) the bit-guessing decoder is nearly as good as the optimal bit decoder:

$$p_{\rm b}^{\rm G} \leq 2 p_{\rm b}. \tag{13.50}$$

(b) the bit-guessing decoder's error probability is related to the block-guessing decoder's by

$$p_{\rm b}^{\rm G} \geq \frac{d_{\min}}{N} p_{\rm B}^{\rm G}. \tag{13.51}$$

Then since $p_{\rm B}^{\rm G} \geq p_{\rm B}$, we have

$$p_{\rm b} > \frac{1}{2} p_{\rm b}^{\rm G} \geq \frac{1}{2} \frac{d_{\min}}{N} p_{\rm B}^{\rm G} \geq \frac{1}{2} \frac{d_{\min}}{N} p_{\rm B}. \tag{13.52}$$

We now prove the two lemmas.

Near-optimality of guessing: Consider first the case of a single bit, with posterior probability $\{p_0, p_1\}$. The optimal bit decoder has probability of error

$$P^{\rm optimal} = \min(p_0, p_1). \tag{13.53}$$

The guessing decoder picks from 0 and 1. The truth is also distributed with the same probability. The probability that the guesser and the truth match is $p_0^2 + p_1^2$; the probability that they mismatch is the guessing error probability,

$$P^{\rm guess} = 2 p_0 p_1 \leq 2 \min(p_0, p_1) = 2 P^{\rm optimal}. \tag{13.54}$$

Since $p_{\rm b}^{\rm G}$ is the average of many such error probabilities, $P^{\rm guess}$, and $p_{\rm b}$ is the average of the corresponding optimal error probabilities, $P^{\rm optimal}$, we obtain the desired relationship (13.50) between $p_{\rm b}^{\rm G}$ and $p_{\rm b}$. □

Relationship between bit error probability and block error probability: The bit-guessing and block-guessing decoders can be combined in a single system: we can draw a sample x_n from the marginal distribution $P(x_n \,|\, \mathbf{r})$ by drawing a sample $(x_n, \mathbf{x})$ from the joint distribution $P(x_n, \mathbf{x} \,|\, \mathbf{r})$, then discarding the value of $\mathbf{x}$.

We can distinguish between two cases: the discarded value of $\mathbf{x}$ is the correct codeword, or not. The probability of bit error for the bit-guessing decoder can then be written as a sum of two terms:

$$\begin{aligned} p_{\rm b}^{\rm G} &= P(\mathbf{x}\text{ correct})P(\text{bit error} \,|\, \mathbf{x}\text{ correct}) \\ &\quad + P(\mathbf{x}\text{ incorrect})P(\text{bit error} \,|\, \mathbf{x}\text{ incorrect}) \\ &= 0 + p_{\rm B}^{\rm G} P(\text{bit error} \,|\, \mathbf{x}\text{ incorrect}). \end{aligned}$$

Now, whenever the guessed $\mathbf{x}$ is incorrect, the true $\mathbf{x}$ must differ from it in at least d bits, so the probability of bit error in these cases is at least d/N. So

$$p_{\rm b}^{\rm G} \geq \frac{d}{N} p_{\rm B}^{\rm G}.$$

QED. □

Solution to exercise 13.20 (p.222). The number of 'typical' noise vectors $\mathbf{n}$ is roughly $2^{NH_2(f)}$. The number of distinct syndromes $\mathbf{z}$ is 2^M. So reliable communication implies

$$M \geq NH_2(f), \tag{13.55}$$

or, in terms of the rate $R = 1 - M/N$,

$$R \leq 1 - H_2(f), \tag{13.56}$$

a bound which agrees precisely with the capacity of the channel.

This argument is turned into a proof in the following chapter.

Solution to exercise 13.24 (p.222). In the three-player case, it is possible for the group to win three-quarters of the time.

Three-quarters of the time, two of the players will have hats of the same colour and the third player's hat will be the opposite colour. The group can win every time this happens by using the following strategy. Each player looks at the other two players' hats. If the two hats are *different* colours, he passes. If they are the *same* colour, the player guesses his own hat is the *opposite* colour.

This way, every time the hat colours are distributed two and one, one player will guess correctly and the others will pass, and the group will win the game. When all the hats are the same colour, however, *all three* players will guess incorrectly and the group will lose.

When any particular player guesses a colour, it is true that there is only a 50:50 chance that their guess is right. The reason that the group wins 75% of the time is that their strategy ensures that when players are guessing wrong, a great many are guessing wrong.

For larger numbers of players, the aim is to ensure that most of the time no one is wrong and occasionally everyone is wrong at once. In the game with 7 players, there is a strategy for which the group wins 7 out of every 8 times they play. In the game with 15 players, the group can win 15 out of 16 times. If you have not figured out these winning strategies for teams of 7 and 15, I recommend thinking about the solution to the three-player game in terms of the locations of the winning and losing states on the three-dimensional hypercube, then thinking laterally.

If the number of players, N, is $2^r - 1$, the optimal strategy can be defined using a Hamming code of length N, and the probability of winning the prize is $N/(N+1)$. Each player is identified with a number $n \in 1 \ldots N$. The two colours are mapped onto `0` and `1`. Any state of their hats can be viewed as a received vector out of a binary channel. A random binary vector of length N is either a codeword of the Hamming code, with probability $1/(N+1)$, or it differs in exactly one bit from a codeword. Each player looks at all the other bits and considers whether his bit can be set to a colour such that the state is a codeword (which can be deduced using the decoder of the Hamming code). If it can, then the player guesses that his hat is the *other* colour. If the state is actually a codeword, all players will guess and will guess wrong. If the state is a non-codeword, only one player will guess, and his guess will be correct. It's quite easy to train seven players to follow the optimal strategy if the cyclic representation of the $(7, 4)$ Hamming code is used (p.19).