CHAPTER 1

THE ORIGIN OF ERROR-CORRECTING CODES

Section 1. An Introduction to Coding

Richard W. Hamming's encounter with the Bell Telephone Laboratories' mechanical relay computer in 1947 (quoted in the Preface) initiated what has come to be known as coding theory. In this chapter we will trace these origins of coding theory and develop enough of the theory itself to prepare for Leech's work in sphere packing which appears in Chapter 2.

Communication is imperfect. Even if a message is accurately stated, it may be garbled during transmission; and the consequences of a mistake in the interpretation of a financial, diplomatic, military or other message may be unfortunate. (Hamming's work was goaded by mistakes which occurred internally in a computer.) As a result, when a message travels from an information source to a destination, both the sender and the receiver would like assurance that the received message either is free of errors or, if it contains errors, those errors will be detectable. Ideally, in case an error is detected, the receiver would like to be able to correct it and recover the original message.

Our interest will lie mainly in the transmission of strings of 0's and 1's of length n, which are called (binary) *n-blocks*. This may, at first glance, seem an artificial restriction. But in fact the entire twenty-six letters of the alphabet can be coded

using 5-blocks since there are thirty-two arrangements of five 0's and 1's. More generally, there are 2^n n-blocks available to form messages.

A sequence of such n-blocks will constitute a *message*. For example, one message of six 3-blocks is

001 101 110 110 010 111.

When errors never occur in transmission, all such n-blocks may be used to form messages without risk of misinterpretation. However, in case errors can occur, the block received may differ from the one sent. One way to compensate for this is to send only blocks that differ from each other so much that one or two errors cannot change one of the selected blocks into another. This restriction requires the use of longer blocks in order to have the same variety of messages possible with the 2^n n-blocks. Of course, this will reduce the amount of information sent per unit time: The gain in reliability is paid for by accepting a slower rate of transmission. A set of selected n-blocks, called *codewords*, will make up a (binary) *code*.

The simplest code has just the two codewords, 0 and 1. In this case no errors in transmission can be detected, since any error changes one codeword into the other. The code {00, 11}, formed by simply repeating the digits of the old codewords, will detect single errors since any single error in a codeword changes it to a noncodeword. For example, 00 could be sent and 01 received. However, two errors in the transmission of a single codeword will go undetected since the received word is also a codeword.

By repeating each of the binary symbols $n-1$ times, we form the *repetition code* consisting of two words of length n, the all-zero word and the all-one word. Up to $n-1$ errors in transmission are detectable. The first digit of each codeword is the *message digit* while the following $n-1$ digits are the *check digits*.

The repetition codes allow the sending of only two different codewords. This restriction can be avoided by using a *block repetition code* whose codewords consist of s copies of all possible r-blocks of binary digits instead of n copies of a single binary digit. There are 2^r words of length $n = rs$ in this code. Take, for example, the case $r = 4$ and $s = 2$. Form the code of sixteen words of length eight where the first four digits coincide with the last four digits. As in the case of the repetition code of length two, all single errors are detectable. For instance, an error in the eighth digit of the codeword 11001100 yields the 8-block 11001101, which is not a codeword. We represent this transmission pictorially as:

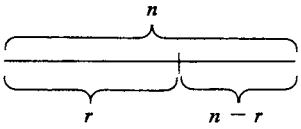
$$11001100 \longrightarrow 11001101$$

(The horizontal arrow is read "is received as.") However, two errors are not always detectable as the following shows:

$$11001100 \longrightarrow 11101110$$

This simple generalization of the repetition codes is equivalent to repeating a given message of r-blocks s times.

A binary block code of length n having r message digits is called an (n, r) *binary block code.* We will refer to it as an (n, r) code. Usually the r message digits occupy the first r positions of a codeword.



A typical codeword.

For example, the repetition code would be an $(n, 1)$ code while the block repetition code would be an (rs, r) code.

Suppose that the probability that a certain message will be

sent is p and that no errors in transmission can occur. If p is close to 1, then no one is "surprised" very much by receiving the message and thus it conveys, in some sense, little information. On the other hand, if p is very small, the "surprise" is much larger and in the same sense, much more information is received. The *information* contained in such a message is given by

$$\log_b\left(\frac{1}{p}\right)$$

for some appropriate base b. This not only agrees with the notion of "surprise," but it is additive in the following sense: If two independent messages occur with positive probabilities p_1 and p_2, then the probability that both occur is the product p_1p_2. In that case, the information sent is

$$\log_b\left(\frac{1}{p_1p_2}\right) = \log_b\left(\frac{1}{p_1}\right) + \log_b\left(\frac{1}{p_2}\right),$$

i.e., the sum of the information of the individual messages. If base 2 logarithms are used, the unit of information is the *bit*. This means that equally likely binary digits carry $\log_2\left(\frac{1}{1/2}\right) =$ 1 bit of information each. The information carried by one codeword of an (n, r) code with w (equally likely) codewords is thus $\log_2\left(\frac{1}{1/w}\right) = \log_2 w$ bits. In case $w = 2^r$, this reduces to r, which is just the number of message digits.

The *information rate* of an (n, r) code with w codewords is the quotient

$$\frac{\log_2 w}{n}.$$

In case $w = 2^r$, this reduces to r/n. This rate records how much information (in bits) is carried on the average per symbol sent. For instance, the information rate of the $(n, 1)$ repetition code is $1/n$, while that of the (rs, r) block repetition

code is r/rs, which is $1/s$. In each case the addition of check digits automatically lowered the information rate.

The goal of coding theory is to devise ways to send messages quickly and accurately. Achieving accuracy over a noisier channel requires more error detection which means more check digits per codeword and thus longer codewords. This slows the flow of information. The cumbersome repetition codes turn out to be very slow but fairly reliable. The concept of "parity" will help realize both goals, speed and reliability, at the same time.

An n-block of binary digits has *even parity* if the sum modulo 2 of the digits is zero (in other words, there are an even number of 1's). Otherwise the n-block has *odd parity*. The summing of the digits is the *parity check*.

With the aid of the parity check, we can devise an efficient single-error-detecting code. For example, to each of the sixteen possible 4-blocks, annex a fifth digit so that the 5-blocks formed have even parity. Thus

$$1100$$

is encoded as

$$11000,$$

while

$$1101$$

becomes

$$11011.$$

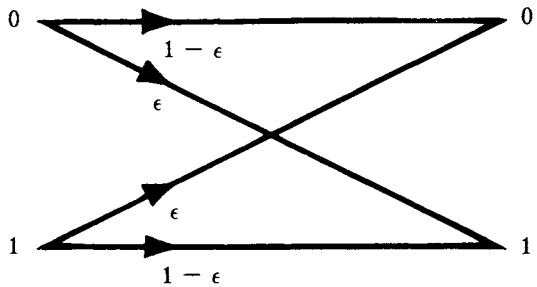
Any single error in the transmission of a 5-block makes the parity of that codeword odd and we say that the parity check *fails*. For example:

$$11000 \longrightarrow 11010$$

indicates an error since $1 + 1 + 0 + 1 + 0 \equiv 1 \pmod 2$. Any

two errors escape detection since the parity of the received word is still even. Like the (8, 4) block repetition code, this (5, 4) single-parity-check code detects single errors. However, its information rate is greater, being $4/5 = 0.80$ as compared with $4/8 = 0.50$ for the (8, 4) code. Since single-error detecting codes with four message digits per codeword must have at least one check digit, the (5, 4) code has the highest rate possible among single-error-detecting codes with four message digits.

Finding high rate codes is not the only problem. One has to consider the channel over which the messages pass. For example, the *binary symmetric channel* is illustrated in Figure 1.1.



ϵ = error probability
(generally $0 \leq \epsilon \leq 1/2$)

FIG. 1.1. The binary symmetric channel.

How much information can pass over this channel? If $\epsilon = 0$, then for one bit of information sent, one could expect to receive one bit. Suppose that $\epsilon > 0$ and that one bit of information is being sent. If an error occurs, then the amount of information lost in that error is $\log_2\left(\frac{1}{\epsilon}\right)$. On the average this is $\epsilon \log_2 \frac{1}{\epsilon}$. Because of the ambiguity ($\epsilon > 0$), even if an error does not occur, $(1 - \epsilon) \log_2\left(\frac{1}{1-\epsilon}\right)$ bits on the average are lost. The

total amount lost is the sum of the two. This means that the amount of information the binary symmetric channel is able to pass or its *channel capacity* (in bits per digit) is given by:

$$C(\epsilon) = 1 - \left[\epsilon \log_2\left(\frac{1}{\epsilon}\right) + (1 - \epsilon)\log_2\left(\frac{1}{1-\epsilon}\right)\right]$$
$$= 1 + \epsilon \log_2 \epsilon + (1 - \epsilon)\log_2(1 - \epsilon).$$

The capacity is a measure, between 0 and 1, of the effect of noise on the channel. It is easy to check that $C(0) = \lim_{\epsilon \to 0} C(\epsilon) = 1$ while $C(1/2) = 0$. The latter simply means that if there is a fifty-fifty chance of error for each binary digit sent, no information can be sent over the channel. Claude E. Shannon showed in 1948 [**108,** p. 411 ff.] that information flows at nearly the rate $C(\epsilon)$ with probability of error arbitrarily small. For binary codes this means that, given arbitrary $\delta > 0$ and $R < C(\epsilon)$, there exist (n, r) codes, with n sufficiently large so that $r/n \geq R$ and the probability of incorrectly decoding a word at the receiving end is less than δ. Unfortunately, no constructive proof exists. It is also known that if $C(\epsilon)$ is exceeded by the code rate, the probability of

TABLE 1.2

Decimal Digit	Codeword
1	11000
2	10100
3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101
0	00011

error can no longer be made arbitrarily small. For further information, see [1], [59] or [85].

In the early 1940's, Bell Telephone Laboratories used a code similar to the (5, 4) single-parity-check code in two of its relay computers [6, p. 349]. The codewords were the ten possible binary 5-blocks with exactly two 1's. These represented the ten decimal digits. Table 1.2 displays the correspondence for this so-called *two-out-of-five code* [58]. Each decimal digit was stored on a group of five relays by having exactly two of them switched on in the proper locations [3, p. 3]. Any single relay malfunction was immediately detectable, since in the case of such a malfunction exactly two relays would no longer operate, but, instead, either one or three.

A later model relay computer used a *three-out-of-five code* [3, p. 3]. This code also has ten codewords since $\binom{5}{3} = 10$. Mathematically, it is obtained from the two-out-of-five code by interchanging the 0's and 1's; physically, it was quite different, since three relays had to be activated instead of two. In case of parity check failure, the checking circuits automatically and immediately halted the machine.

We now turn from error detection to error correction. It is one thing to know that a particular message is wrong, but it is quite another to be able to reconstruct the correct message from the wrong message. To study this problem, we shall take the geometric point of view introduced by Hamming in his 1950 paper on error-correcting codes [54, pp. 154–156]. He first considered the unit cube in Euclidean n-dimensional space, E^n, whose vertices are the 2^n n-tuples of 0's and 1's. A binary code with words of length n is then a certain subset of the vertices of this cube. The darkened circles in Figure 1.3 illustrate a code with four codewords of length three. The four darkened vertices are the codewords 000, 011, 101 and 110 formed by the addition of a parity check digit to each pair of binary digits. This (3, 2) code detects single errors. Figure 1.4 shows the three possible 3-blocks formed by a single error

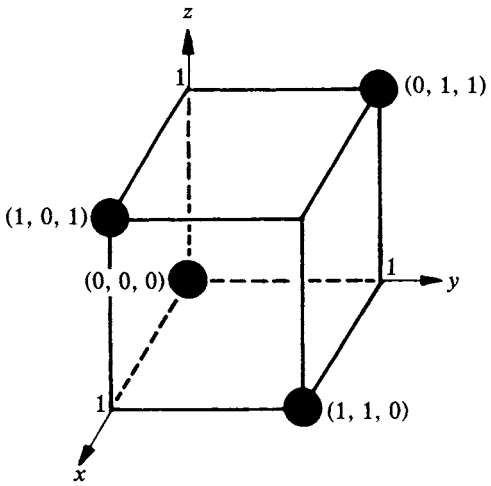


FIG. 1.3

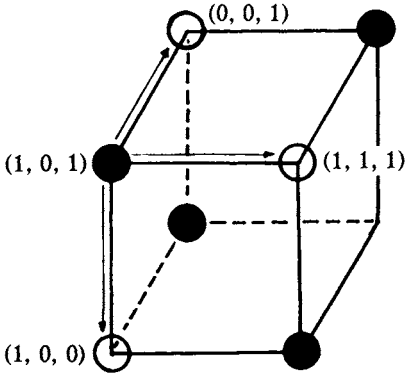


FIG. 1.4

in the transmission of the codeword 101. This figure also shows why two errors in a codeword are not detectable: a second error sends the result of the first error to another codeword. One error in a codeword changes one coordinate and two errors change two coordinates; for any positive integer e, e errors alter e coordinates. This observation of Hamming motivated his definition of the *distance*, D, between two n-blocks of binary digits as the number of coordinates in which the corresponding vertices differ. For instance, $D(101, 011) = 2$, while $D(101, 010) = 3$. In terms of the geometry of the cube, this distance is just the number of edges in a shortest path between the two vertices. This function D, called the *Hamming distance*, a standard notion in coding theory, satisfies the usual definition of a metric. For n-blocks x, y and z,

$$D(x, y) \geq 0 \text{ and } D(x, y) = 0 \Leftrightarrow x = y$$
$$D(x, y) = D(y, x)$$
and $$D(x, z) \leq D(x, y) + D(y, z).$$

The *minimum distance* of a code is the minimum of all the distances between two nonidentical codewords of the code. For the (3, 2) single-parity-check code shown in Figure 1.3, it is two.

While single-error detection is possible with a code whose minimum distance is two, single-error correction is only possible with a code whose minimum distance is at least three. One example is the (3, 1) repetition code pictured in Figure 1.5. (We will assume that any code contains the all-zero codeword.) Figure 1.6 shows the words that can be received if exactly one error is made in the transmission of the codeword 000. Each of the three possible errors still is a distance two from the codeword 111. If a maximum of one error is assumed to occur in the transmission of a single codeword, then the error can be found and we can figure out exactly which codeword has been sent. However, the correcting ability disappears if two errors occur in the transmission of one