Mathematics and Statistics · Carleton University

70.385 NOTES ON ERROR CORRECTING CODES

1 Introduction

Whenever information is transmitted (for example, by radio from a satellite or across telephone lines) or stored (for example, on your computer's hard disk or on a CD), there is the possibility that the information will be *corrupted* so that what is received or read at a later time is not the same as the original information. Error correcting codes can often minimize this problem, by enabling us to reconstruct the correct information even after errors have occurred.

The basic idea of these codes is to use *redundancy.* This means that the information sent is repeated in certain subtle ways. Everyday speech uses a great deal of redundancy. This is illustrated by the fact that if you go to a noisy party you may still understand what someone is saying even though you miss hearing many of the words. There is enough redundancy in English (and other languages) that often a fraction of the words (or letters) is enough to convey the meaning. Fr xmpl, t s pssbl t rd ths sntnc whr ll th vwls r mssng.

REMARK Many computer files contain considerable redundancy. There are programs which can recognize this redundancy and remove it (these *compress* the files), for example, files with extensions like ZIP or GIF have been compressed in this way. It is sometimes possible to reduce the length of a file considerably in such a way that the compressed files can later be *uncompressed* to their original form. However, this natural redundancy is not useful for error correction because it fails to be present uniformly through the files.

Do not confuse error correcting codes with codes used to conceal information. The latter are formally called systems of *encryption.* For example, encryption is used to prevent an eavesdropper from reading personal information transmitted over the Internet and for scrambling TV signals so that only subscribers who have paid for a descrambler can receive them, as well as for preserving military and diplomatic secrets. The design of secure *encryption* systems is called *cryptography.* Cryptography also involves some interesting mathematics, but its object and methods are quite different from the objects and methods of error correction.

2 Error detecting codes

To clarify the discussion we shall describe our codes in the following way. A *sender* S wishes to send a message to a *receiver* R. The message passes along a *noisy channel* which may *corrupt* it by modifying it in various ways, and the message received may not be the message sent. S therefore *encodes* the message before transmitting it, so that R will receive a (possibly corrupted) encoded message. When an error correcting code is used R expects, even if the received message is corrupted, to be able to *decode* the message and obtain the original message of S. In practice there may be occasional cases where the message is so badly corrupted that it cannot be properly decoded, but the code will be designed so that this happens very rarely.

We start our discussion with something which is a little simpler to describe than error correcting codes. These are codes for *error detection.* They are designed to signal when an error has occurred, but do not attempt to correct the error. In practice, if a message is transmitted using an error detecting code and the receiver finds that there is an error in the transmission, then the receiver must ask the sender to re-send the message. The situation can be diagrammatically described by:

$$\boxed{\text{original message}} \to \boxed{\text{message encoded}} \to \boxed{\text{message transmitted along noisy channel}}$$
$$\to \boxed{\text{encoded (possibly corrupted) message received}} \to \boxed{\text{decoding and error detection}}$$

where the process is repeated for a given message string until no error is detected.

EXAMPLE 1 *A commonly used error detection code is based on message strings consisting of seven bits ($0's$ and $1's$), say $b_1 b_2 ... b_7$. The encoded string $b_1 b_2 ... b_8$ is obtained by adjoining an additonal bit b_8 where b_8 is 0 or 1 according to whether the sum of the other digits is even or odd. In other words, $b_8 \equiv b_1 + ... + b_7 \pmod 2$. If exactly one of the eight bits is changed in transmission, then the receiver will be able to recognize that an error has occurred because this relationship between the bits no longer holds. However, if two errors (or any even number of errors) occur the receiver will not realize that there has been an error. The added bit b_8 is called a* check-sum *bit.*

REMARK In many situations it is reasonable to assume that multiple errors occur infrequently, and that single errors occur much more often than multiple errors. Most codes are therefore designed to be able to detect or correct when one (or a small number) or errors have occurred in a string of particular length. This is the situation which we shall discuss here. Sometimes this is not a good assumption: there may be *bursts* of errors which all come at once.

EXAMPLE 2 *Another commonly used error correcting code is the code used for the ISBN (International Standard Book Number) numbers which appear at the front of every book that is published. This is a 10 digit number $d_1 d_2 ... d_{10}$ which uniquely identifies the book. The first nine digit are from $\{0, 1, 2, ..., 9\}$ and the last digit is either from this set or is equal to X (which denotes 10). The last digit is a check digit and is chosen so that*

$$d_{10} \equiv 1d_1 + 2d_2 + ... + 9d_9 \pmod{11} \ \textit{or, equivalently,}$$
$$1d_1 + 2d_2 + ... + 10d_{10} \equiv 0 \pmod{11}$$

The advantage of this particular error detecting code is that it (i) detects an error if exactly one of the digits is wrong, (ii) detects an error if two of the digits has been interchanged, and (iii) will detect an error in many cases when two or more digits are incorrect.

Various kinds of more elaborate check sums are used for computer files, and can detect when a file has become corrupted by a physical defect in a chip or a disc, or the file has been affected by a virus.

3 Error correcting codes

Although error detecting codes are useful, it is often better to have codes which can automatically correct themselves. There is a wide literature on error correcting codes. We shall only be looking at some elementary examples.

In what follows we shall make the following assumptions which are quite realistic in many cases.

- We shall assume that our messages (or data) consist of strings of *bits* (denoted as $0's$ and $1's$). We shall suppose that these strings are broken up into blocks of fixed length, say m. A message (or file of data) will usually consist of a great many of these blocks. For example, for a text, such as the manuscript for these notes, each letter and punctuation symbol is translated into a special sequence of eight bits according to a scheme called the ASCII character set, and the manuscript is stored on my computer as this string of bits.

- Our codes will be constructed by taking each of these blocks and adding k *check bits* to obtain a new block of total length $n := m + k$. These new blocks will be called *codewords*, and the m original bits in each block are called the message bits or *information bits.* An encoded message will usually consist of many of these codewords, but it will be enough to deal with each codeword separately. The check bits are the redundancy which we are adding, and this redundancy will be used for error correction.

- We shall assume that the errors which occur when the codewords are transmitted are *bit errors*: some of the bits may change (from 0 to 1 or from 1 to 0) but the number of bits does not change. We shall also assume that the errors which occur will generally occur one at a time (not in bursts). More particularly, for some chosen value of d, we shall suppose that with high probability at most d of the n bits of a codeword received are in error, and ignore the possibility that more than d errors occur (see the Remark below). For example, if $d = 1$, then we shall be designing our code so that it can correct a codeword if a single error occurs in it; the correction scheme may fail in the event that more than one error occurs.

REMARK Consider the (possibly corrupted) codewords which are received. In the absence of "bursts" of errors, it is often reasonable to suppose that for each bit there is the same (small) probability p that an error has occurred in transmission, and that the probability that one particular bit is incorrect is independent of the probability of another particular bit being wrong. In this situation the number of bit errors in a particular codeword is given by the *binomial distribution.* Suppose that the probability of an error in each particular bit is p. Then the probability that we have k errors in a codeword of length n is $p_k := \binom{n}{k}p^k(1-p)^k$ for $k = 0, 1, ..., n$.

For example, if on the average there is one bit error in every 10000 bits, then the probability of a particular bit being in error is $p = 1/10000 = 0.0001$. If the codewords have length $n = 10$, then we find that $p_0 = \binom{10}{0}p^0(1-p)^{10} = 0.9990004...$, $p_1 = \binom{10}{1}p^1(1-p)^9 = 0.0009991...$, $p_2 = \binom{10}{2}p^2(1-p)^2 = 0.0000004...$ and the other p_k are negligible. Thus, if w codewords are transmitted, then the probability that all of the received words each have at most one error is $(p_0 + p_1)^w = 0.9999995^w$. For example, if our message consists of $w = 10^5$ codewords, then this probability is 0.95. In other words, in this situation, about 95% of the time a single error correcting code will succeed in correcting all of the 10^5 codewords.

EXAMPLE 3 *A very simple example of a code is a* repetition code. *In this code the first bit is the only information bit, and all the other bits (check bits) are equal to this first bit. For example, if $n = 2$, then there are the two codewords:* 00 *and* 11. *This code will* detect *one error, but it is not able to* correct *an error; if you receive* 01 *then you know that an error has occurred (since* 01 *is not a codeword), but you cannot tell which bit is wrong. A better situation occurs when $n = 3$ ($m = 1$ and $k = 2$). Again there are only two codewords:* 000 *and* 111, *but in this case if a single error occurs you can not only recognize the error but also see what the correct codeword is (so $d = 1$). More generally, for any $d \geq 1$ with $n = 2d + 1$ and the two codewords* 000...0 *and* 111...1 *we can correct up to d errors for any $d \geq 1$.*

The repetition code shows that it is possible to design codes which can correct any number of errors through simple repetition of the message. However, these codes are not very efficient. To encode a message using the repetition code of length n each block of length $m = 1$ in the original message is replaced by a block of length n, so that the message which is transmitted is n times as long as the original message.

The two aims in designing an error correcting code are

- to be able to correct any d of errors which may occur in codewords of given length n (in general, we should like d to be as large as possible, but often $d = 1$ is satisfactory); and

- to encode the message economically in the sense that the *rate* $m/n = m/(m + k)$ is as large as possible (it is always ≤ 1).

Unfortunately, these two objectives are mutually incompatible. If we want to correct more errors per codeword, then we need more check bits and fewer information bits; this will decrease the rate m/n. A well-designed code is a compromise being able to correct multiple errors and having a code with a good rate. The choice of design depends on how many errors we anticipate and the lowest rate that we can tolerate.

REMARK Your favourite musical CD is a storage device for a binary file containing more than 10^{10} bits. About one third of the information on the disc is non-audio information which is used to process the music before you hear it, and to ensure that the sound is virtually perfect. The data is stored in codewords of length 588. Only 192 bits of each codeword contain the audio information and 64 of the bits are check bits. This enables a very high level of error correction, and explains why even a badly mistreated CD may still sound fine. (How much do you have to scratch your CD before it sounds bad?) [*Ref.* K.C. Pohlmann, "Principles of Digital Audio" (2nd. ed.), Howard W. Sams, 1989.]

4 Linear codes

In order to be able to apply algebraic techniques to our design of codes we shall represent the bit values 0 and 1 of our codewords as elements of the field $\mathbb{Z}_2 = \{0, 1\}$. Recall that in $\mathbb{Z}_2$ arithmetic is done "modulo 2", so addition and multiplication are defined by:

$$0+0=1+1=0,\ 0+1=1+0=1 \ \text{ and } \ 0\cdot 0=0\cdot 1=1\cdot 0=0,\ 1\cdot 1=1$$

Now $(\mathbb{Z}_2)^n$ is the vector space of dimension n over the field $\mathbb{Z}_2$ consisting of all n-tuples (rows) from $\mathbb{Z}_2$. We shall find it convenient to use notations like $[0,1,0,0]$ and 0100 interchangeably to represent vectors from these spaces.

The error correcting codes described above consist of codewords of the following form. Each codeword has m information bits, say $b_1, ..., b_m$, and k check bits, say $b_{m+1}, ..., b_{m+k}$ where the check bits depend in some way on the information bits and have been introduced to enable us to correct the codeword if one or more error occurs. In principle, the check bits could be any functions of the information bits (or each other), but in practice it has turned out to be simplest to design good codes where each check bit is a sum (in $\mathbb{Z}_2$!) of one or more of the information bits. Note that the number of codewords is equal to the number of distinct message strings $b_1 b_2 ... b_m$, namely 2^m.

EXAMPLE 4 *Consider the case where $m = 3$ and $k = 3$ where we define $b_4 = b_1 + b_2$, $b_5 = b_2 + b_3$ and $b_6 = b_3 + b_1$. This code has eight ($= 2^3$) codewords:*

000000	001011	010110	011101
100101	101110	110011	111000

A little study should convince you that this code can correct single errors. In other words, if only one bit is changed in one of these codewords then from the corrupted codeword you can decide what the correct word was. Later we shall give a systematic way to do this. Note that this code has rate $3/6 = 1/2$.

The set C of codewords in the last example forms a vector subspace of $(\mathbb{Z}_2)^6$. The easiest way to see this is to note that C is just the set of all solutions of the set of homogeneous linear equations $b_4 = b_1 + b_2$, $b_5 = b_2 + b_3$ and $b_6 = b_3 + b_1$ which define the code. Indeed there is a natural basis for the vector space C chosen so that the information bits form the standard basis for $(\mathbb{Z}_2)^3$ This leads to the following definition.

Definition *A* systematic (n, m)-linear code *of length n is a subspace C of dimension m in $(\mathbb{Z}_2)^n$ which has a basis $v_1, ..., v_m$ with the property that the first m entries of v_j are all $0'$s except that the jth entry is 1. We call the $m \times n$ matrix G with rows $v_1, ..., v_m$ the* generating matrix *for C.*

EXAMPLE 5 *For the code C defined in Example 4 we construct the generating matrix as follows. Since $m = 3$, we consider the three messages* 100, 010 *and* 001 *corresponding to the standard basis for $(\mathbb{Z}_2)^3$. These are encoded, respectively, by adding check bits to get the codewords:* 100101, 010110 *and* 001011. *The generating matrix for C is therefore*

$$G := \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Now for any message $[b_1, b_2, b_3]$ we have

$$\begin{aligned} [b_1, b_2, b_3]G = [b_1, b_2, b_3] &\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \\ &= [b_1, b_2, b_3, (b_1 + b_2), (b_2 + b_3), (b_1 + b_3)] \end{aligned}$$

which is clearly the code word corresponding to $[b_1, b_2, b_3]$. Thus multiplying by the generating matrix gives precisely the same codewords as those described earlier in terms of check bits (Example 4)

In general, if C is a systematic (n, m)-code with generating matrix G, then G can be written in block form $G = \begin{bmatrix} I_m & A \end{bmatrix}$ where I_m is the $m \times m$ identity matrix and A is some $m \times (n - m)$ matrix. (In the previous example $A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$.).

As Example 5 illustrates, the generating matrix enables us to encode message strings by a simple matrix multiplication. If the message is the vector $[b_1, b_2, ..., b_m]$, then the corresponding codeword is $[b_1, b_2, ..., b_m]G$. Because of the special form of G we find that this product is a vector of the form $[b_1, b_2, ..., b_m, b_{m+1}, ..., b_n]$; so the first m components are the same as the components of the message vector and the remaining components are linear combinations of $b_1, b_2, ..., b_m$ with coefficients (0 or 1) determined by the entries in A. In other words, $b_1, b_2, ..., b_m$ are the information bits in the codeword and $b_{m+1}, ..., b_n$ are check bits. A number of vectors in a systematic (n, m)-linear code C is equal to the number of ways in which the m information bits can be chosen, so $|C| = 2^m$.

REMARK It is possible to define general (n, m)-*linear codes* where, by definition, every subspace C of dimension m in $(\mathbb{Z}_2)^n$ is an (n, m)-linear code. For such general linear codes, a generating matrix G is simply any $m \times n$ matrix with entries in $\mathbb{Z}_2$ whose rows form a basis for the subspace C. However, in what follows we shall restrict our attention to *systematic* codes where we have chosen the generating matrix for C in the echelon reduced form described above.

5 Limitations of Decoding

In the last section we discussed systematic codes and how to use the generating matrix to encode a message string. We now turn to the problem of decoding. This is more complicated because we want our decoding scheme to decode not only correctly transmitted codewords, but also codewords which have been corrupted (in up to d bits if our code is designed to correct up to d errors). We first consider some of the natural limitations which occur. To do this it is helpful to use a suitable "distance" function.

Definition *We define the* Hamming distance *between to vectors u, v in $(\mathbb{Z}_2)^n$ to be equal to the number of components in u which differ from the corresponding component in v. This distance is denoted $d(u, v)$. The* weight *of a vector u is simply the number of nonzero components of u and is written $wt(u)$.*

It is easily seen that $wt(u) = d(u,0)$ (where 0 denotes the vector with all components 0) and that $d(u,v) = d(v,u) = wt(u-v)$.

EXAMPLE 6 *In $(\mathbb{Z}_2)^5$ we have $wt(01001) = wt(11000) = 2$ and $d(01101, 10001) = wt(11100) = 3$ (note that $01101 - 10001 = 11100$ as vectors over $\mathbb{Z}_2$).*

A useful property of the Hamming distance is that $d(u,v) \leq d(u,w) + d(w,v)$ for all u, v, w (this is called the *triangle inequality* - can you see why?). Indeed, if u and v differ in their ith components, then either u and w differ in their ith components or w and v differ in their ith components. Hence the number of components where u and v differ is not greater than the number where u and w differ plus the number where w and v differ.

We can now state a criterion which allows us to calculate the maximum number of errors which a particular linear code can correct.

Theorem 1 *Let C be an (n,m)-linear code, and let d be the largest integer such that $wt(v) \geq 2d+1$ for all nonzero vectors v in C. Then*
(a) there is a decoding scheme which can correct up to d errors in a code word; and
(b) there is no decoding scheme which can consistently correct more than d errors.

PROOF. (a) We define our decoder D as a function which applies to every vector $w \in (\mathbb{Z}_2)^n$ to give a vector $D(w) \in C$. The vector $D(w)$ is defined to be the codeword u such that the Hamming distance $d(D(w), u)$ is as small as possible (if this distance is achieved by more than one codeword u then we pick one of these codewords). Now suppose that a codeword v is transmitted and the received word is w. If at most d errors occurred, then $d(v,w) \leq d$.

We claim that $D(w) = v$ and so D gives an accurate decoding. To prove this we have to show that $d(w,v) < d(w,u)$ for every $u \neq v$ in C. Suppose, on the contrary that there exists some $u \neq v$ in C such that $d(w,u) \leq d(w,v)$. Then the triangle inequality shows that

$$wt(u-v) = d(u,v) \leq d(u,w) + d(w,v) = d(u,w) + d(v,w) \leq 2d(v,w) \leq 2d.$$

Since C is a subspace of $(\mathbb{Z}_2)^n$, and u, v are distinct elements of C, therefore $u - v$ is a nonzero element of C. Since we have shown that $wt(u-v) \leq 2d$, this contradicts the choice of the integer d. Thus $D(w) = v$. This shows that D will correctly decode received words which contain up to d errors.

(b) On the other hand, the hypothesis on d shows that there exists a nonzero code word u such that $d(u,0) = wt(u) \leq 2d+2$. If we change $d+1$ of the $1's$ in u to $0's$ then we obtain a vector w such that $d(u,w) = d+1$, and $d(0,w) = wt(w) \leq d+1$. Thus w could be the received word when the codeword u is transmitted and $d+1$ bit errors occur in transmission, and it also could be the received word when the code word 0 is transmitted and $wt(w) \leq d+1$ errors occur in transmission. Since there is no way for the receiver to distinguish between these two possibilities when w is received, there is no decoding scheme which can consistently correct up to $d+1$ errors. ■

EXAMPLE 7 *Consider the code C in Example 4. The minimum weight of every nonzero codeword is 3, and so the theorem above shows that C can be used to correct single errors, but cannot consistently correct double errors. For example, if the receiver receives the word 000011 this word differs the codeword 001011 in one bit, and from each of the codewords 000000 and 110011 in two bits. If the receiver only knows that at most two bits are in error, there is no way to decide which of these three codewords was transmitted. On the other hand, if it is assumed that at most one bit is in error, then 000011 should be decoded as 001011.*

The following theorem gives an upper bound on the number m of information bits which a code of length n can have if it is able to correct single, double or multiple errors.

Theorem 2 *Suppose that C is an (n, m)-linear code. Then*
(a) if C can correct all single errors, then $1 + n \leq 2^{n-m}$;
(b) if C can correct both single and double errors, then $1 + n + \binom{n}{2} \leq 2^{n-m}$;
(c) if C can correct up to d errors, then $1 + n + \binom{n}{2} + ... + \binom{n}{d} \leq 2^{n-m}$.

PROOF. An (n, m)-code C consists of 2^m vectors from the vector space $(\mathbb{Z}_2)^n$ which contains a total of 2^n vectors.

(a) We are supposing that C can correct single errors. For each codeword v there are exactly $1+n$ vectors w such that $d(v, w) \leq 1$, namely, v itself and the n vectors obtained by changing exactly one of the n components of v. Call this set of vectors $B_1(v)$. Since C can correct single errors, then it is necessary that none of the vectors w lies in more than one of the sets $B_1(v)$. Therefore $B_1(u) \cap B_1(v) = \emptyset$ whenever u and v are distinct codewords. Since each $|B_1(v)| = 1 + n$, and there are only 2^n vectors altogether, this shows that

$$2^n \geq \sum_{v \in C} |B_1(v)| = (1 + n)\,|C| = (1 + n)2^m$$

and so $2^{n-m} \geq 1 + n$ as required.

(b) Now suppose that C can detect both single and double errors. Then we can correctly decode any codeword v after it has been corrupted to a vector w where $d(v, w) \leq 2$. Let $B_2(v)$ be the set of w satisfying $d(v, w) \leq 2$. Then $B_2(v)$ contains v, the n vectors obtained from v by changing exactly one of its components and the $\binom{n}{2}$ vectors obtained from v by changing exactly two of its components. Hence $|B_2(v)| = 1 + n + \binom{n}{2}$. Again, because C can correct all single and double errors, we must have $B_2(u) \cap B_2(v) = \emptyset$ whenever u and v are distinct codewords. Thus, as before, we get

$$2^n \geq \sum_{v \in C} |B_2(v)| = (1 + n + \binom{n}{2})\,|C| = (1 + n + \binom{n}{2})2^m$$

and the result follows.

(c) The proof of the general result is similar. ■

The following table illustrates the bounds of Theorem 2 for single and double error correcting codes. For example, if $n = 10$, then part (a) of the theorem shows that $11 \leq 2^{10-m}$ and so the number m of information bits is at most 6 ($11 \leq 2^{10-6}$ but $11 \not\leq 2^{10-7}$). This gives the entry 6 in the table for single errors under $n = 10$ and shows that the maximum rate for a code of length 10 which corrects single errors is $\leq 6/10$. We note that Theorem 2 does not guarantee the existence of a code of length 10 with this rate. One of the challenges of coding theory is to design codes with the best possible rate for a given length and given error correcting capabilities

n	3	4	5	6	7	8	9	10	11	12
single errors	1	1	2	3	4	4	5	6	7	8
double errors	0	0	1	1	2	2	3	4	4	5

Bounds on m for single and double error correction

6 Decoding a systematic linear code

The proof of Theorem 1(a) describes, in principle, the best way to decode: if w is the received word, then we should decode w by $D(w)$ where $D(w)$ is a codeword v such that $d(v, w)$ is as small as possible. Commonly, however, a code may contain a great many codewords ($|C| = 2^m$ where m is the number of information bits), and in such cases it is not very efficient to decode by searching through the set of all code words to find one which minimizes the Hamming distance. A more effective way of decoding a systematic linear code is described in this section.

Recall that a systematic (n,m)-code C can be described by an $m \times n$ generating matrix G of the form $[I_m \; A]$ where the block I_m represents the $m \times m$ identity matrix and A is an $m \times (n-m)$ matrix representing the sums involved in the $(n-m)$ check bits (see the paragraph following Example 5). Consider the $n \times m$ matrix H of the form $\begin{bmatrix} A \\ I_{n-m} \end{bmatrix}$. Then

$$GH = \begin{bmatrix} I_m & A \end{bmatrix} \begin{bmatrix} A \\ I_{n-m} \end{bmatrix} = [A + A] = 0_{m,n-m}$$

where $0_{m,n-m}$ represents the $m \times (n-m)$ zero matrix. (We have used the fact that $a + a = 0$ for all $a \in \mathbb{Z}_2$.) Since the codewords in C are all of the form $u = sG$ where $s \in (\mathbb{Z}_2)^m$, it follows that $uH = sGH = s0_{m,n-m} = 0$ for every codeword u (where 0 denotes the row vector consisting of $n - m$ $0's$.) Because the matrix H has rank $n - m$ it can also be shown that $uH \neq 0$ if $u \in (\mathbb{Z}_2)^n$ is not a codeword. This leads to the following definition.

Definition *For any systematic (n,m)-code C with generating matrix G, the matrix H defined above is called the* parity check *matrix. For each vector $w \in (\mathbb{Z}_2)^n$, the* syndrome *of w is the $(n-m)$-row vector wH. The syndrome is zero if and only if $w \in C$. (*Syndrome *is a medical term which means a pattern of symptoms which characterize a medical condition or disease.)*

EXAMPLE 8 *Consider Example 5. In this case*

$$H := \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The syndrome of the vector $w := [0,1,1,1,0,0]$ is $[0,1,1,1,0,0]H = [0,0,1]$ (remember you are working in $\mathbb{Z}_2$!). Since this syndrome is not the zero vector, w is not a codeword.

Now consider the case where a codeword v is transmitted and a (possibly corrupted) word w is received. Write $e := w - v$ for the error in w. Note that e is a vector in $(\mathbb{Z}_2)^n$ which has 1 in each component where w differs from v and 0 in the other components. In particular, $wt(e) = d(v,w)$ so, if at most d bits in w are in error, then $wt(e) \leq d$. Since $w = v + e$, the syndrome of w is $wH = (v + e)H = vH + eH = eH$ because $vH = 0$ for each codeword v. In particular, if $d = 1$ (so we are correcting single errors), then we assume that e has at most one entry equal to 1. If e has its ith entry equal to 1, then $wH = eH$ is equal to the ith row of H. This leads to the following theorem.

Theorem 3 *Let C be a systematic (n,m)-linear code, and suppose that the parity check matrix H has all its rows different and that no row consists entirely of $0's$. Then C can correct (at least) single errors. If a received word w has only one error then it can be decoded by computing the syndrome $s := wH$. If s is the zero vector, then w is the required code word. If s is nonzero, then it must equal one of the rows of H. If s equals the ith row, then the error in w is in the ith entry.*

EXAMPLE 9 *Consider the vector w in Example 8. Since its syndrome is nonzero it is not a code word. However, the syndrome is equal to the 6th row of H, and so the error (assuming that there* *is only one) in w is in the 6th entry. Hence w is corrected to $[0,1,1,1,0,1]$ (check that this is a codeword). Since the code C is a systematic $(6,3)$-linear code, the original message consists of the first three entries of this codeword. Thus w is decoded as $[0,1,1]$. On the other hand, suppose that we received $w' := [1,1,1,1,1,1]$. Then the syndrome $s' := w'H = [1,1,1]$. This is nonzero and is not equal to any row of H. Therefore w' could not have been obtained from any codeword with a single error; at least two bit errors must have occurred.*

A theorem similar to Theorem 3 but rather more complicated explains the form of a code which can correct both single and double errors ($d = 2$).

Theorem 4 *Let C be a systematic (n,m)-linear code, and let H be its parity check matrix. Construct a list L consisting of the rows of H and all $\binom{n}{2}$ vectors which are obtained as the sum of two rows of H. Suppose that all of the vectors on this list are different and that none is the zero vector. Then C can correct all single and double errors as follows. If w is the received word and w differs in at most two entries from the codeword which is sent, then the syndrome $s := wH$ is either 0 (and w is correct) or s is one of the vectors on the list L. If s is one of the rows of H, say the ith row, then w has a single error in the ith entry. Otherwise s is the sum of two rows of H, say the ith and the jth, and in this case w has an error in both the ith and the jth entries.*

EXAMPLE 10 *We shall show how to design a systematic $(8,2)$-code which can correct up to two errors. The parity check matrix must be of the form*

$$H := \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where the first two rows are chosen so that the conditions of the theorem are met. A little experimentation shows that these two rows must each have at least four $1'$s. In fact we can choose the two rows to be $[1\,1\,1\,1\;0\,0]$ and $[0\,0\,1\,1\,1\,1]$. The corresponding generating matrix is

$$G := \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Now the message $[1\;0]$ is encoded by $[1\,0]G = [1\,0\,1\,1\,1\,1\,0\,0]$. If this vector is transmitted with two errors, say in the 2nd and 5th positions, then the received vector is $[1\,1\,1\,1\,0\,1\,0\,0]$. Now the syndrome is

$$s := [1\,1\,1\,1\,0\,1\,0\,0]H = [0\,0\,0\,1\,1\,1]$$

Since s is equal to the sum of the 2nd and 5th rows of H, this tells us (correctly) that the errors in the received vector are in these two positions. Thus we find the correct code word $[1\,0\,1\,1\,1\,1\,0\,0]$ and from that find the original message (= the first two entries in the code word).

7 Problems

1. Using the error detecting code describe in Example 1 decide which of the following words have detectable errors: 11001001, 01100100, 11111100, 11010011.

2. Which of the following are valid ISBNs: 0-521-65378-9, 0-521-36664-X, 0-534-19002-8, 0-12-751955-6?

3. Decode the following words using the $(3,1)$-code of Example 3: 101, 111, 011, 001, 000.

4. Consider the linear code C' defined in terms of the generating matrix

$$G := \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

   (a) Write down all the code words in C' and give the message strings which they encode.

   (b) Can C' detect all single errors? Can it detect all double errors? [Explain]

   (c) Can C' correct all single errors? Can it correct all double errors? [Explain]

   (d) To what codewords should the following received words be decoded? 01001, 10110, 01000, 01011.

5. Consider the linear code C defined by the parity check matrix

$$H := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   Find the set of codewords and minimum distance for C. Is C able to correct all single errors?

6. Consider a binary $(8,4)$-code C whose code words $b_1 b_2 ... b_8$ satisfy the conditions that the first two rows and first two columns of the matrix

$$\begin{bmatrix} b_1 & b_2 & b_5 \\ b_3 & b_4 & b_6 \\ b_7 & b_8 & \end{bmatrix}$$

   sum to 0 in $\mathbb{Z}_2$.

   (a) Describe the code words in terms of information bits and check bits. Show that the code is systematic. How many code words are there?

   (b) Write down the generating matrix G for this code.

   (c) Encode the message strings: 1011, 0101 and 1000 directly from the definition and also by using the generating matrix.

7. *(continuing the previous exercise)*

   (a) Write down the parity check matrix H for the code C.

   (b) Calculate the syndromes of the following received words: 11001011, 00011111, 01101111 and 01011010. Which of these words contain errors?

(c) Assuming that the received words in (b) are results of at most single errors in codewords, find the corrected codewords and the original message strings.

8. Consider the $(7,4)$-code whose parity check matrix has every possible nonzero vector of length 3 as a row ([1]):

$$H := \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Find the generating matrix and describe the encoding process in terms of check bits calculated from information bits.

(b) Show that the code C' can correct all single errors.

(c) Show that there is no other linear code of length 7 with more information bits.

9. Generalize the previous example to show how to construct systematic $(2^r-1, 2^r-r-1)$-codes which can correct all single errors (for $r = 1, 2, 3, \ldots$). Show that there are no linear codes of length $2^r - 1$ with more information bits which can correct all single errors.

10. Give a careful proof of the claim in Example 10 that for the parity check matrix given there the rows and the sums of every pair of rows are all different. You should not have to write out all the sums explicitly–use arguments based on the weights of the rows and their sums.

11. Write down all the codewords in Example 10 and use Theorem 1 to give another proof that this code can correct all single and double errors.

12. Show that there is no systematic $(7,2)$-code which can correct all single and double errors.

13. Consider the systematic linear code C of length 12 with information bits $b_1, \ldots, b_6$ and check bits $b_7, \ldots, b_{12}$ where the latter are defined by the conditions that all rows and columns of the following matrix sum to 0 in $\mathbb{Z}_2$:

$$\begin{bmatrix} b_1 & b_2 & b_3 & b_7 \\ b_4 & b_5 & b_6 & b_8 \\ b_9 & b_{10} & b_{11} & b_{12} \end{bmatrix}$$

Calculate the generating matrix and the parity check matrix, and show that C can correct all single errors. Can it correct all double errors?

(JDD 2000.10)

[1]This code is the first error correcting codes ever described. It was invented in 1947 by R.W. Hamming and is know as the $(7,4)$-Hamming code. Two years later M.J.E. Golay constructed an amazing $(23,12)$-(binary) code which corrects up to three errors, as well as a $(11,6)$-(ternary) code which corrects up to two errors.