Chapter 24 Error Correcting Codes

In a good cryptographic system, changing one bit in the ciphertext changes enough bits in the corresponding plaintext to make it unreadable. Therefore, we need a way of detecting and correcting errors that could occur when ciphertext is transmitted.

Many noncryptographic situations also require error correction; for example, fax machines, computer hard drives, CD players, and anything that works with digitally represented data. Error correcting codes solve this problem.

Though coding theory (communication over noisy channels) is technically not part of cryptology (communication over nonsecure channels), in Section 24.10 we describe how error correcting codes can be used to construct a public key cryptosystem.

24.1 Introduction

All communication channels contain some degree of noise, namely interference caused by various sources such as neighboring channels, electric impulses, deterioration of the equipment, etc. This noise can interfere with data transmission. Just as holding a conversation in a noisy room becomes more difficult as the noise becomes louder, so too does data transmission become more difficult as the communication channel becomes noisier. In order to hold a conversation in a loud room, you either raise your voice, or you are forced to repeat yourself. The second method is the one that will concern us; namely, we need to add some redundancy to the transmission in order for the recipient to be able to reconstruct the message. In the following, we give several examples of techniques that can be used. In each case, the symbols in the original message are replaced by *codewords* that have some redundancy built into them.

Example 1. (repetition codes)

Consider an alphabet $\{A, B, C, D\}$. We want to send a letter across a noisy channel that has a probability $p = 0.1$ of error. If we want to send C, for example, then there is a 90% chance that the symbol received is C. This leaves too large a chance of error. Instead, we repeat the symbol three times, thus sending CCC. Suppose an error occurs and the received word is CBC. We take the symbol that occurs most frequently as the message, namely C. The probability of the correct message being found is the probability that all three letters are correct plus the probability that exactly one of the three letters is wrong:

$$(0.9)^3 + 3(0.9)^2(0.1) = 0.972,$$

which leaves a significantly smaller chance of error.

Two of the most important concepts for codes are error detection and error correction. If there are at most two errors, this repetition code allows us to detect that errors have occurred. If the received message is CBC, then there could be either one error from CCC or two errors from BBB; we cannot tell which. If at most one error has occurred, then we can correct the error and deduce that the message was CCC. Note that if we used only two repetitions instead of three, we could detect the existence of one error, but we could not correct it (did CB come from BB or CC?).

This example was chosen to point out that error correcting codes can use arbitrary sets of symbols. Typically, however, the symbols that are used are mathematical objects such as integers mod a prime or binary strings. For example, we can replace the letters A, B, C, D by 2-bit strings: 00, 01, 10, 11. The preceding procedure (repeating three times) then gives us the codewords

$$000000,\ 010101,\ 101010,\ 111111.$$

Example 2. (parity check)

Suppose we want to send a message of seven bits. Add an eighth bit so that the number of nonzero bits is even. For example, the message 0110010 becomes 01100101, and the message 1100110 becomes 11001100. An error of one bit during transmission is immediately discovered since the message received will have an odd number of nonzero bits. However, it is impossible to tell which bit is incorrect, since an error in any bit could have yielded the odd number of nonzero bits. When an error is detected, the best thing to do is resend the message.

Example 3. (two-dimensional parity code)

The parity check code of the previous example can be used to design a code that can correct an error of one bit. The two-dimensional parity code arranges the data into a two-dimensional array, and then parity bits are computed along each row and column.

To demonstrate the code, suppose we want to encode the 20 data bits 10011011001100101011. We arrange the bits into a 4×5 matrix

$$\begin{matrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{matrix}$$

and calculate the parity bits along the rows and columns. We define the last bit in the lower right corner of the extended matrix by calculating the parity of the parity bits that were calculated along the columns. This results in the 5×6 matrix

$$\begin{matrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1. \end{matrix}$$

Suppose that this extended matrix of bits is transmitted and that a bit error occurs at the bit in the third row and fourth column. The receiver arranges the received bits into a 5×6 matrix and obtains

$$\begin{matrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1. \end{matrix}$$

The parities of the third row and fourth column are odd, so this locates the error as occurring at the third row and

fourth column.

If two errors occur, this code can detect their existence. For example, if bit errors occur at the second and third bits of the second row, then the parity checks of the second and third columns will indicate the existence of two bit errors. However, in this case it is not possible to correct the errors, since there are several possible locations for them. For example, if the second and third bits of the fifth row were incorrect instead, then the parity checks would be the same as when these errors occurred in the second row.

Example 4. (Hamming [7, 4] code)

The original message consists of blocks of four binary bits. These are replaced by codewords, which are blocks of seven bits, by multiplying (mod 2) on the right by the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

For example, the message 1100 becomes

$$(1,\ 1,\ 0,\ 0) \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \equiv (1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1) \pmod 2.$$

Since the first four columns of G are the identity matrix, the first four entries of the output are the original message. The remaining three bits provide the redundancy that allows error detection and correction. In fact, as we'll see, we can easily correct an error if it affects only one of the seven bits in a codeword.

Suppose, for example, that the codeword 1100011 is sent but is received as 1100001. How do we detect and correct the error? Write G in the form $[I_4,\, P]$, where P is a 4×3 matrix. Form the matrix $H = [P^T,\, I_3]$, where P^T is the transpose of P. Multiply the message received times the transpose of H:

$$(1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1)\begin{pmatrix}1&1&0&1&1&0&0\\1&0&1&1&0&1&0\\0&1&1&1&0&0&1\end{pmatrix}^T$$

$$\equiv (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 1)\begin{pmatrix}1&1&0\\1&0&1\\0&1&1\\1&1&1\\1&0&0\\0&1&0\\0&0&1\end{pmatrix} \equiv (0,\ 1,\ 0) \pmod 2.$$

This is the 6th row of H^T, which means there was an error in the 6th bit of the message received. Therefore, the correct codeword was 1100011. The first four bits give the original message 1100. If there had been no errors, the result of multiplying by H^T would have been $(0,\ 0,\ 0)$, so we would have recognized that no correction was needed. This rather mysterious procedure will be explained when we discuss Hamming codes in Section 24.5. For the moment, note that it allows us to correct errors of one bit fairly efficiently.

The Hamming [7, 4] code is a significant improvement over the repetition code. In the Hamming code, if we want to send four bits of information, we transmit seven bits. Up to two errors can be detected and up to one error can be corrected. For a repetition code to achieve this level of error detection and correction, we need to transmit 12 bits in order to send a 4-bit message. Later, we'll express this mathematically by saying that the code rate of this Hamming code is 4/7, while the code rate of the repetition code is $4/12 = 1/3$. Generally, a higher code rate is better, as long as not too much error correcting capability is lost. For example, sending a 4-bit

message as itself has a code rate of 1 but is unsatisfactory in most situations since there is no error correction capability.

Example 5. (ISBN code)

The International Standard Book Number (ISBN) provides another example of an error detecting code. The ISBN is a 10-digit codeword that is assigned to each book when it is published. For example, the first edition of this book had ISBN number 0-13-061814-4. The first digit represents the language that is used; 0 indicates English. The next two digits represent the publisher. For example, 13 is associated with Pearson/Prentice Hall. The next six numbers correspond to a book identity number that is assigned by the publisher. The tenth digit is chosen so that the ISBN number $a_1 a_2 \cdots a_{10}$ satisfies

$$\sum_{j=1}^{10} j a_j \equiv 0 \pmod{11}.$$

Notice that the equation is done modulo 11. The first nine numbers $a_1 a_2 \cdots a_9$ are taken from $\{0, 1, \cdots 9\}$ but a_{10} may be 10, in which case it is represented by the symbol X. Books published in 2007 or later use a 13-digit ISBN, which uses a slightly different sum and works mod 10.

Suppose that the ISBN number $a_1 a_2 \cdots a_{10}$ is sent over a noisy channel, or is written on a book order form, and is received as $x_1 x_2 \cdots x_{10}$. The ISBN code can detect a single error, or a double error that occurs due to the transposition of the digits. To accomplish this, the receiver calculates the weighted checksum

$$S = \sum_{j=1}^{10} j x_j \pmod{11}.$$

If $S \equiv 0 \pmod{11}$, then we do not detect any errors, though there is a small chance that an error occurred and was undetected. Otherwise, we have detected an error. However, we cannot correct it (see Exercise 2).

If $x_1 x_2 \cdots x_{10}$ is the same as $a_1 a_2 \cdots a_{10}$ except in one place x_k, we may write $x_k = a_k + e$ where $e \neq 0$. Calculating S gives

$$S = \sum_{j=1}^{10} j a_j + ke \equiv ke \pmod{11}.$$

Thus, if a single error occurs, we can detect it. The other type of error that can be *reliably* detected is when a_k and a_l have been transposed. This is one of the most common errors that occur when someone is copying numbers. In this case $x_l = a_k$ and $x_k = a_l$. Calculating S gives

$$\begin{aligned} S = \sum_{j=1}^{10} j x_j &= \sum_{j=1}^{10} j a_j + (k-l)a_l + (l-k)a_k \pmod{11} \\ &\equiv (k-l)(a_l - a_k) \pmod{11} \end{aligned}$$

If $a_l \neq a_k$, then the checksum is not equal to 0, and an error is detected.

Example 6. (Hadamard code)

This code was used by the *Mariner* spacecraft in 1969 as it sent pictures back to Earth. There are 64 codewords; 32 are represented by the rows of the 32×32 matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & -1 & \cdots & -1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & -1 & -1 & 1 & \cdots & -1 \end{pmatrix}.$$

The matrix is constructed as follows. Number the rows and columns from 0 to 31. To obtain the entry h_{ij} in the

ith row and jth column, write $i = a_4a_3a_2a_1a_0$ and $j = b_4b_3b_2b_1b_0$ in binary. Then

$$h_{ij} = (-1)^{a_0b_0+a_1b_1+\cdots+a_4b_4}.$$

For example, when $i = 31$ and $j = 3$, we have $i = 11111$ and $j = 00011$. Therefore, $h_{31,\,3} = (-1)^2 = 1$.

The other 32 codewords are obtained by using the rows of $-H$. Note that the dot product of any two rows of H is 0, unless the two rows are equal, in which case the dot product is 32.

When *Mariner* sent a picture, each pixel had a darkness given by a 6-bit number. This was changed to one of the 64 codewords and transmitted. A received message (that is, a string of 1s and -1s of length 32) can be decoded (that is, corrected to a codeword) as follows. Take the dot product of the message with each row of H. If the message is correct, it will have dot product 0 with all rows except one, and it will have dot product ± 32 with that row. If the dot product is 32, the codeword is that row of H. If it is -32, the codeword is the corresponding row of $-H$. If the message has one error, the dot products will all be ± 2, except for one, which will be ± 30. This again gives the correct row of H or $-H$. If there are two errors, the dot products will all be $0,\ \pm 2,\ \pm 4$, except for one, which will be $\pm 32,\ \pm 30$, or ± 28. Continuing, we see that if there are seven errors, all the dot products will be between -14 and 14, except for one between -30 and -16 or between 16 and 30, which yields the correct codeword. With eight or more errors, the dot products start overlapping, so correction is not possible. However, detection is possible for up to 15 errors, since it takes 16 errors to change one codeword to another.

This code has a relatively low code rate of $6/32$, since it uses 32 bits to send a 6-bit message. However, this is

balanced by a high error correction rate. Since the messages from *Mariner* were fairly weak, the potential for errors was high, so high error correction capability was needed. The other option would have been to increase the strength of the signal and use a code with a higher code rate and less error correction. The transmission would have taken less time and therefore potentially have used less energy. However, in this case, it turned out that using a weaker signal more than offset the loss in speed. This issue (technically known as **coding gain**) is an important engineering consideration in the choice of which code to use in a given application.

24.2 Error Correcting Codes

A sender starts with a message and **encodes** it to obtain codewords consisting of sequences of symbols. These are transmitted over a noisy channel, depicted in Figure 24.1, to the receiver. Often the sequences of symbols that are received contain errors and therefore might not be codewords. The receiver must **decode**, which means correct the errors in order to change what is received back to codewords and then recover the original message.

Figure 24.1 Encoding and Decoding

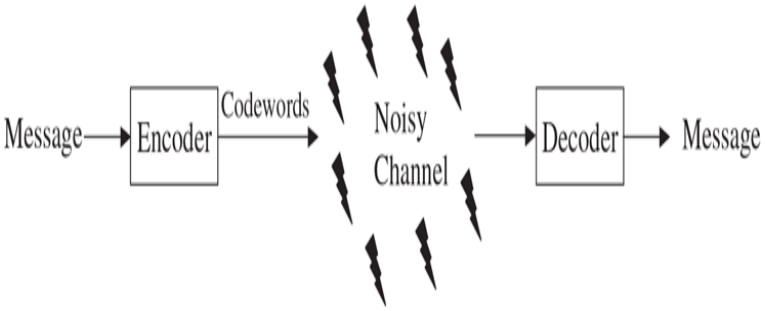

Figure 24.1 Full Alternative Text

The symbols used to construct the codewords belong to an alphabet. When the alphabet consists of the binary bits 0 and 1, the code is called a **binary code**. A code that uses sequences of three symbols, often represented as integers mod 3, is called a **ternary code**. In general, a code that uses an alphabet consisting of q symbols is called a ***q*-ary code**.

Definition

Let A be an alphabet and let A^n denote the set of n-tuples of elements of A. A **code of length n** is a nonempty subset of A^n.

The n-tuples that make up a code are called **codewords**, or code vectors. For example, in a binary repetition code where each symbol is repeated three times, the alphabet is the set $A = \{0, 1\}$ and the code is the set $\{(0, 0, 0), (1, 1, 1)\} \subset A^3$.

Strictly speaking, the codes in the definition are called **block codes**. Other codes exist where the codewords can have varying lengths. These will be mentioned briefly at the end of this chapter, but otherwise we focus only on block codes.

For a code that is a random subset of A^n, decoding could be a time-consuming procedure. Therefore, most useful codes are subsets of A^n satisfying additional conditions. The most common is to require that A be a finite field, so that A^n is a vector space, and require that the code be a subspace of this vector space. Such codes are called *linear* and will be discussed in Section 24.4.

For the rest of this section, however, we work with arbitrary, possibly nonlinear, codes. We always assume that our codewords are n-dimensional vectors.

In order to decode, it will be useful to put a measure on how close two vectors are to each other. This is provided by the Hamming distance. Let u, v be two vectors in A^n. The **Hamming distance** $d(u, v)$ is the number of places where the two vectors differ. For example, if we use binary vectors and have the vectors
$u = (1, 0, 1, 0, 1, 0, 1, 0)$ and
$v = (1, 0, 1, 1, 1, 0, 0, 0)$, then u and v differ in two places (the 4th and the 7th) so $d(u, v) = 2$. As another example, suppose we are working with the usual English alphabet. Then $d(fourth, eighth) = 4$ since the two strings differ in four places.

The importance of the Hamming distance $d(u, v)$ is that it measures the minimum number of "errors" needed for u to be changed to v. The following gives some of its basic properties.

Proposition

$d(u, v)$ is a metric on A^n, which means that it satisfies

1. $d(u, v) \geq 0$, and $d(u, v) = 0$ if and only if $u = v$
2. $d(u, v) = d(v, u)$ for all u, v
3. $d(u, v) \leq d(u, w) + d(w, v)$ for all u, v, w.

The third property is often called the triangle inequality.

Proof. (1) $d(u, v) = 0$ is exactly the same as saying that u and v differ in no places, which means that $u = v$. Part (2) is obvious. For part (3), observe that if u and v differ in a place, then either u and w differ at that place, or v and w differ at that place, or both. Therefore, the number of places where u and v differ is less than or equal to the number of places where u and w differ, plus the number of places where v and w differ.

For a code C, one can calculate the Hamming distance between any two distinct codewords. Out of this table of distances, there is a minimum value $d(C)$, which is called the **minimum distance** of C. In other words,

$$d\left(C\right) = \min\left\{d\left(u,\ v\right) | u,\ v\ \in\ C,\ u \neq v\right\}.$$

The minimum distance of C is very important number, since it gives the smallest number of errors needed to change one codeword into another.

When a codeword is transmitted over a noisy channel, errors are introduced into some of the entries of the vector. We correct these errors by finding the codeword whose Hamming distance from the received vector is as

small as possible. In other words, we change the received vector to a codeword by changing the fewest places possible. This is called **nearest neighbor decoding**.

We say that a code can **detect** up to s errors if changing a codeword in at most s places cannot change it to another codeword. The code can **correct** up to t errors if, whenever changes are made at t or fewer places in a codeword c, then the closest codeword is still c. This definition says nothing about an efficient algorithm for correcting the errors. It simply requires that nearest neighbor decoding gives the correct answer when there are at most t errors. An important result from the theory of error correcting codes is the following.

Theorem

1. A code C can detect up to s errors if $d(C) \geq s + 1$. 2. A code C can correct up to t errors if $d(C) \geq 2t + 1$.

Proof.

1. Suppose that $d(C) \geq s + 1$. If a codeword c is sent and s or fewer errors occur, then the received message r cannot be a different codeword. Hence, an error is detected.

2. Suppose that $d(C) \geq 2t + 1$. Assume that the codeword c is sent and the received word r has t or fewer errors; that is, $d(c, r) \leq t$. If c_1 is any other codeword besides c, we claim that $d(c_1, r) \geq t + 1$. To see this, suppose that $d(c_1, r) \leq t$. Then, by applying the triangle inequality, we have

   $$2t + 1 \leq d(C) \leq d(c, c_1) \leq d(c, r) + d(c_1, r) \leq t + t = 2t.$$

   This is a contradiction, so $d(c_1, r) \geq t + 1$. Since r has t or fewer errors, nearest neighbor decoding successfully decodes r to c.

How does one find the nearest neighbor? One way is to calculate the distance between the received message r and each of the codewords, then select the codeword with the smallest Hamming distance. In practice, this is impractical for large codes. In general, the problem of

decoding is challenging, and considerable research effort is devoted to looking for fast decoding algorithms. In later sections, we'll discuss a few decoding techniques that have been developed for special classes of codes.

Before continuing, it is convenient to introduce some notation.

Notation

A code of length n, with M codewords, and with minimum distance $d = d(C)$, is called an **(n, M, d) code**.

When we discuss linear codes, we'll have a similar notation, namely, an $[n, k, d]$ code. Note that this latter notation uses square brackets, while the present one uses curved parentheses. (These two similar notations cause less confusion than one might expect!) The relation is that an $[n, k, d]$ binary linear code is an $(n, 2^k, d)$ code.

The binary repetition code $\{(0, 0, 0), (1, 1, 1)\}$ is a $(3, 2, 3)$ code. The Hadamard code of Exercise 6, Section 24.1, is a $(32, 64, 16)$ code (it could correct up to 7 errors because $16 \geq 2 \cdot 7 + 1$).

If we have a q-ary (n, M, d) code, then we define the **code rate**, or **information rate**, R by

$$R = \frac{\log_q M}{n}.$$

For example, for the Hadamard code, $R = \log_2(64)/32 = 6/32$. The code rate represents the ratio of the number of input data symbols to the number of transmitted code symbols. It is an important parameter to consider when implementing real-world systems, as it represents what fraction of the bandwidth

is being used to transmit actual data. The code rate was mentioned in Examples 4 and 6 in Section 24.1. A few limitations on the code rate will be discussed in Section 24.3.

Given a code, it is possible to construct other codes that are essentially the same. Suppose that we have a codeword c that is expressed as $c = (c_1, c_1, \cdots, c_n)$. Then we may define a positional permutation of c by permuting the order of the entries of c. For example, the new vector $c' = (c_2, c_3, c_1)$ is a positional permutation of $c = (c_1, c_2, c_3)$. Another type of operation that can be done is a symbol permutation. Suppose that we have a permutation of the q-ary symbols. Then we may fix a position and apply this permutation of symbols to that fixed position for every codeword. For example, suppose that we have the following permutation of the ternary symbols $\{0 \to 2, 1 \to 0, 2 \to 1\}$, and that we have the following codewords: $(0, 1, 2)$, $(0, 2, 1)$, and $(2, 0, 1)$. Then applying the permutation to the second position of all of the codewords gives the following vectors: $(0, 0, 2)$, $(0, 1, 1)$, and $(2, 2, 1)$.

Formally, we say that two codes are **equivalent** if one code can be obtained from the other by a series of the following operations:

1. Permuting the positions of the code
2. Permuting the symbols appearing in a fixed position of all codewords

It is easy to see that all codes equivalent to an (n, M, d) code are also (n, M, d) codes. However, for certain choices of n, M, d, there can be several inequivalent (n, M, d) codes.

24.3 Bounds on General Codes

We have shown that an (n, M, d) code can correct t errors if $d \geq 2t + 1$. Hence, we would like the minimum distance d to be large so that we can correct as many errors as possible. But we also would like for M to be large so that the code rate R will be as close to 1 as possible. This would allow us to use bandwidth efficiently when transmitting messages over noisy channels. Unfortunately, increasing d tends to increase n or decrease M.

In this section, we study the restrictions on n, M, and d without worrying about practical aspects such as whether the codes with good parameters have efficient decoding algorithms. It is still useful to have results such as the ones we'll discuss since they give us some idea of how good an actual code is, compared to the theoretical limits.

First, we treat upper bounds for M in terms of n and d. Then we show that there exist codes with M larger than certain lower bounds. Finally, we see how some of our examples compare with these bounds.

24.3.1 Upper Bounds

Our first result was given by R. Singleton in 1964 and is known as the **Singleton bound**.

Theorem

Let C be a q-ary (n, M, d) code. Then

$$M \leq q^{n-d+1}.$$

Proof. For a codeword $c = (a_1, \ldots, a_n)$, let $c' = (a_d, \ldots, a_n)$. If $c_1 \neq c_2$ are two codewords, then they differ in at least d places. Since ${c_1}'$ and ${c_2}'$ are obtained by removing $d-1$ entries from c_1 and c_2, they must differ in at least one place, so ${c_1}' \neq {c_2}'$. Therefore, the number M of codewords c equals the number of vectors c' obtained in this way. There are at most q^{n-d+1} vectors c' since there are $n-d+1$ positions in these vectors. This implies that $M \leq q^{n-d+1}$, as desired.

Corollary

The code rate of a q-ary (n, M, d) code is at most $1 - \frac{d-1}{n}$.

Proof. The corollary follows immediately from the definition of code rate.

The corollary implies that if the **relative minimum distance** d/n is large, the code rate is forced to be small.

A code that satisfies the Singleton bound with equality is called an **MDS code** (maximum distance separable). The Singleton bound can be rewritten as $q^d \leq q^{n+1}/M$, so an MDS code has the largest possible value of d for a given n and M. The Reed-Solomon codes (Section 24.9) are an important class of MDS codes.

Before deriving another upper bound, we need to introduce a geometric interpretation that is useful in error correction. A **Hamming sphere** of radius t centered at a codeword c is denoted by $B(c, t)$ and is defined to be all vectors that are at most a Hamming

distance of t from the codeword c. That is, a vector u belongs to the Hamming sphere $B(c, t)$ if $d(c, u) \leq t$. We calculate the number of vectors in $B(c, t)$ in the following lemma.

Lemma

A sphere $B(c, r)$ in n-dimensional q-ary space has

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r$$

elements.

Proof. First we calculate the number of vectors that are a distance 1 from c. These vectors are the ones that differ from c in exactly one location. There are n possible locations and $q-1$ ways to make an entry different. Thus the number of vectors that have a Hamming distance of 1 from c is $n(q-1)$. Now let's calculate the number of vectors that have Hamming distance m from c. There are $\binom{n}{m}$ ways in which we can choose m locations to differ from the values of c. For each of these m locations, there are $q-1$ choices for symbols different from the corresponding symbol from c. Hence, there are

$$\binom{n}{m}(q-1)^m$$

vectors that have a Hamming distance of m from c. Including the vector c itself, and using the identity $\binom{n}{0} = 1$, we get the result:

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r.$$

We may now state the **Hamming bound**, which is also called the **sphere packing bound**.

Theorem

Let C be a q-ary (n, M, d) code with $d \geq 2t + 1$. Then

$$M \leq \frac{q^n}{\sum_{j=0}^{t} \binom{n}{j} (q-1)^j}.$$

Proof. Around each codeword c we place a Hamming sphere of radius t. Since the minimum distance of the code is $d \geq 2t + 1$, these spheres do not overlap. The total number of vectors in all of the Hamming spheres cannot be greater than q^n. Thus, we get

$$\begin{aligned}&(\text{number of codewords}) \times (\text{number of elements per sphere}) \\ &= M \sum_{j=0}^{t} \binom{n}{j} (q-1)^j \leq q^n.\end{aligned}$$

This yields the desired inequality for M.

An (n, M, d) code with $d = 2t + 1$ that satisfies the Hamming bound with equality is called a **perfect code**. A perfect t-error correcting code is one such that the M Hamming spheres of radius t with centers at the codewords cover the entire space of q-ary n-tuples. The Hamming codes (Section 24.5) and the Golay code G_{23} (Section 24.6) are perfect. Other examples of perfect codes are the trivial $(n, q^n, 1)$ code obtained by taking all n-tuples, and the binary repetition codes of odd length (Exercise 15).

Perfect codes have been studied a lot, and they are interesting from many viewpoints. The complete list of perfect codes is now known. It includes the preceding examples, plus a ternary $[11, 6, 5]$ code constructed by Golay. We leave the reader a caveat. A name like *perfect codes* might lead one to assume that perfect codes are the best error correcting codes. This, however, is not true, as there are error correcting codes, such as Reed-Solomon codes, that are not perfect codes yet have better

error correcting capabilities for certain situations than perfect codes.

24.3.2 Lower Bounds

One of the problems central to the theory of error correcting codes is to find the largest code of a given length and given minimum distance d. This leads to the following definition.

Definition

Let the alphabet A have q elements. Given n and d with $d \leq n$, the largest M such that an (n, M, d) code exists is denoted $A_q(n, d)$.

We can always find at least one (n, M, d) code: Fix an element a_0 of A. Let C be the set of all vectors $(a, a, \ldots, a, a_0, \ldots, a_0)$ (with d copies of a and $n - d$ copies of a_0) with $a \in A$. There are q such vectors, and they are at distance d from each other, so we have an (n, q, d) code. This gives the trivial lower bound $A_q(n, d) \geq q$. We'll obtain much better bounds later.

It is easy to see that $A_q(n, 1) = q^n$: When a code has minimum distance $d = 1$, we can take the code to be all q-ary n-tuples. At the other extreme, $A_q(n, n) = q$ (Exercise 7).

The following lower bound, known as the **Gilbert-Varshamov bound**, was discovered in the 1950s.

Theorem

Given n, d with $n \geq d$, there exists a q-ary (n, M, d) code with

$$M \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j}.$$

This means that

$$A_q(n, d) \geq \frac{q^n}{\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j}.$$

Proof. Start with a vector c_1 and remove all vectors in A^n (where A is an alphabet with q symbols) that are in a Hamming sphere of radius $d-1$ about that vector. Now choose another vector c_2 from those that remain. Since all vectors with distance at most $d-1$ from c_1 have been removed, $d(c_2, c_1) \geq d$. Now remove all vectors that have distance at most $d-1$ from c_2, and choose c_3 from those that remain. We cannot have $d(c_3, c_1) \leq d-1$ or $d(c_3, c_2) \leq d-1$, since all vectors satisfying these inequalities have been removed. Therefore, $d(c_3, c_i) \geq d$ for $i = 1, 2$. Continuing in this way, choose $c_4, c_5, \ldots$, until there are no more vectors.

The selection of a vector removes at most

$$\sum_{j=0}^{d-1} \binom{n}{j} (q-1)^j$$

vectors from the space. If we have chosen M vectors $c_1, \ldots, c_M$, then we have removed at most

$$M \sum_{j=1}^{d-1} \binom{n}{j} (q-1)^j$$

vectors, by the preceding lemma. We can continue until all q^n vectors are removed, which means we can continue at least until

$$M \sum_{j=1}^{d-1} \binom{n}{j} (q-1)^j \geq q^n.$$

Therefore, there exists a code $\{c_1, \ldots, c_M\}$ with M satisfying the preceding inequality.

Since $A_q(n, d)$ is the largest such M, it also satisfies the inequality.

There is one minor technicality that should be mentioned. We actually have constructed an (n, M, e) code with $e \geq d$. However, by modifying a few entries of c_2 if necessary, we can arrange that $d(c_2, c_1) = d$. The remaining vectors are then chosen by the above procedure. This produces a code where the minimal distance is exactly d.

If we want to send codewords with n bits over a noisy channel, and there is a probability p that any given bit will be corrupted, then we expect the number of errors to be approximately pn when n is large. Therefore, we need an (n, M, d) code with $d > 2pn$. We therefore need to consider (n, M, d) codes with $d/n \approx x > 0$, for some given $x > 0$. How does this affect M and the code rate?

Here is what happens. Fix q and choose x with $0 < x < 1 - 1/q$. The **asymptotic Gilbert-Varshamov bound** says that there is a sequence of q-ary (n, M, d) codes with $n \to \infty$ and $d/n \to x$ such that the code rate approaches a limit $\geq H_q(x)$, where

$$H_q(x) = 1 - x \log_q (q-1) + x \log_q (x) + (1-x) \log_q (1-x).$$

The graph of $H_2(x)$ is as in Figure 24.2. Of course, we would like to have codes with high error correction (that is, high x), and with high code rate ($= k/n$). The asymptotic result says that there are codes with error correction and code rate good enough to lie arbitrarily close to, or above, the graph.

Figure 24.2 The Graph of $H_2(x)$

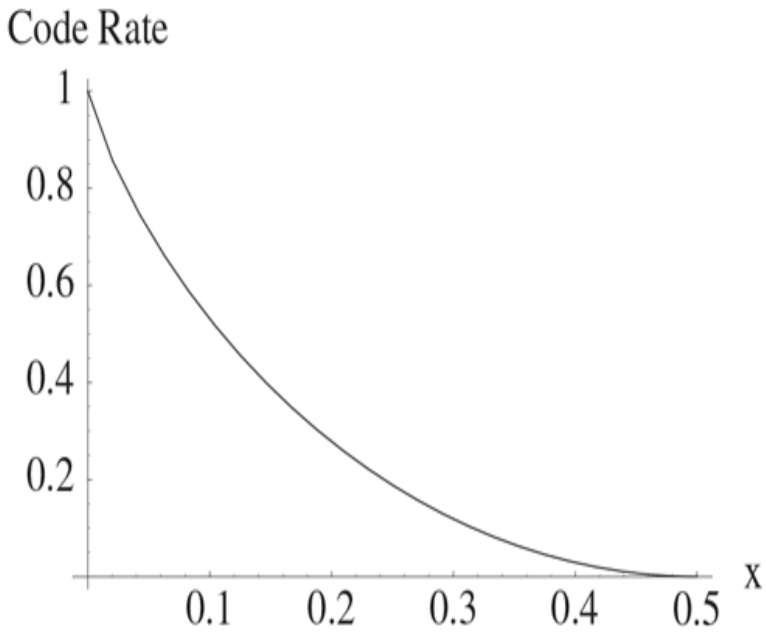

Figure 24.2 Full Alternative Text

The existence of certain sequences of codes having code rate limit strictly larger than $H_q(x)$ (for certain x and q) was proved in 1982 by Tsfasman, Vladut, and Zink using Goppa codes arising from algebraic geometry.

Examples

Consider the binary repetition code C of length 3 with the two vectors $(0, 0, 0)$ and $(1, 1, 1)$. It is a $(3, 2, 3)$ code. The Singleton bound says that $2 = M \leq 2$, so C is an MDS code. The Hamming bound says that

$$2 = M \leq \frac{2^3}{\binom{3}{0} + \binom{3}{1}} = 2,$$

so C is also perfect. The Gilbert-Varshamov bound says that there exists a $(3, M, 3)$ binary code with

$$M \geq \frac{2^3}{\binom{3}{0} + \binom{3}{1} + \binom{3}{2}} = \frac{8}{7},$$

which means $M \geq 2$.

The Hamming $[7, 4]$ code has $M = 16$ and $d = 3$, so it is a $(7, 16, 3)$ code. The Singleton bound says that $16 = M \leq 2^5$, so it is not an MDS code. The Hamming bound says that

$$16 = M \leq \frac{2^7}{\binom{7}{0} + \binom{7}{1}} = 16,$$

so the code is perfect. The Gilbert-Varshamov bound says that there exists a $(7, M, 3)$ code with

$$M \geq \frac{2^7}{\binom{7}{0} + \binom{7}{1} + \binom{7}{2}} = \frac{128}{29} \approx 4.4,$$

so the Hamming code is much better than this lower bound. Codes that have efficient error correction algorithms and also exceed the Gilbert-Varshamov bound are currently relatively rare.

The Hadamard code from Section 24.1 is a binary (because there are two symbols) $(32, 64, 16)$ code. The Singleton bound says that $64 = M \leq 2^{17}$, so it is not very sharp in this case. The Hamming bound says that

$$64 = M \leq \frac{2^{32}}{\sum_{j=0}^{7} \binom{32}{j}} \approx 951.3.$$

The Gilbert-Varshamov bound says there exists a binary $(32, M, 16)$ code with

$$M \geq \frac{2^{32}}{\sum_{j=0}^{15} \binom{32}{j}} \approx 2.3.$$

24.4 Linear Codes

When you are having a conversation with a friend over a cellular phone, your voice is turned into digital data that has an error correcting code applied to it before it is sent. When your friend receives the data, the errors in transmission must be accounted for by decoding the error correcting code. Only after decoding are the data turned into sound that represents your voice.

The amount of delay it takes for a packet of data to be decoded is critical in such an application. If it took several seconds, then the delay would become aggravating and make holding a conversation difficult.

The problem of efficiently decoding a code is therefore of critical importance. In order to decode quickly, it is helpful to have some structure in the code rather than taking the code to be a random subset of A^n. This is one of the primary reasons for studying linear codes. For the remainder of this chapter, we restrict our attention to linear codes.

Henceforth, the alphabet A will be a finite field $\mathbf{F}$. For an introduction to finite fields, see Section 3.11. For much of what we do, the reader can assume that $\mathbf{F}$ is $\mathbf{Z}_2 = \{0, 1\}$ = the integers mod 2, in which case we are working with binary vectors. Another concrete example of a finite field is $\mathbf{Z}_p$ = the integers mod a prime p. For other examples, see Section 3.11. In particular, as is pointed out there, $\mathbf{F}$ must be one of the finite fields $GF(q)$; but the present notation is more compact. Since we are working with arbitrary finite fields, we'll use "=" instead of "$\equiv$" in our equations. If you want to think of $\mathbf{F}$ as being $\mathbf{Z}_2$, just replace all equalities between elements of $\mathbf{F}$ with congruences mod 2.

The set of n-dimensional vectors with entries in $\mathbf{F}$ is denoted by $\mathbf{F}^n$. They form a vector space over $\mathbf{F}$. Recall that a subspace of $\mathbf{F}^n$ is a nonempty subset S that is closed under linear combinations, which means that if s_1, s_2 are in S and a_1, a_2 are in $\mathbf{F}$, then $a_1 s_1 + a_2 s_2 \in S$. By taking $a_1 = a_2 = 0$, for example, we see that $(0, 0, , \ \ldots, 0) \in S$.

Definition

A **linear code** of dimension k and length n over a field $\mathbf{F}$ is a k-dimensional subspace of $\mathbf{F}^n$. Such a code is called an $\boldsymbol{[n, k]}$ **code**. If the minimum distance of the code is d, then the code is called an $\boldsymbol{[n, k, d]}$ **code**.

When $\mathbf{F} = \mathbf{Z}_2$, the definition can be given more simply. A binary code of length n and dimension k is a set of 2^k binary n-tuples (the codewords) such that the sum of any two codewords is always a codeword.

Many of the codes we have met are linear codes. For example, the binary repetition code $\{(0, 0, 0), (1, 1, 1)\}$ is a one-dimensional subspace of $\mathbf{Z}_2^3$. The parity check code from Exercise 2 in Section 24.1 is a linear code of dimension 7 and length 8. It consists of those binary vectors of length 8 such that the sum of the entries is 0 mod 2. It is not hard to show that the set of such vectors forms a subspace. The vectors

$$(1, 0, 0, 0, 0, 0, 0, 1), \ (0, 1, 0, 0, 0, 0, 0, 1), \ \ \ldots, (0, 0, 0, 0, 0, 0, 1, 1)$$

form a basis of this subspace. Since there are seven basis vectors, the subspace is seven-dimensional.

The Hamming [7, 4] code from Example 4 of Section 24.1 is a linear code of dimension 4 and length 7. Every codeword is a linear combination of the four rows of the matrix G. Since these four rows span the code and are linearly independent, they form a basis.

The ISBN code (Example 5 of Section 24.1) is not linear. It consists of a set of 10-dimensional vectors with entries in $\mathbf{Z}_{11}$. However, it is not closed under linear combinations since X is not allowed as one of the first nine entries.

Let C be a linear code of dimension k over a field $\mathbf{F}$. If $\mathbf{F}$ has q elements, then C has q^k elements. This may be seen as follows. There is a basis of C with k elements; call them $v_1, \ldots, v_k$. Every element of C can be written uniquely in the form $a_1 v_1 + \cdots + a_k v_k$, with $a_1, \ldots, a_k \in \mathbf{F}$. There are q choices for each a_i and there are k numbers a_i. This means there are q^k elements of C, as claimed. Therefore, an $[n, k, d]$ linear code is an (n, q^k, d) code in the notation of Section 24.2.

For an arbitrary, possibly nonlinear, code, computing the minimum distance could require computing $d(u, v)$ for every pair of codewords. For a linear code, the computation is much easier. Define the **Hamming weight** $wt(u)$ of a vector u to be the number of nonzero places in u. It equals $d(u, 0)$, where 0 denotes the vector $(0, 0, \ldots, 0)$.

Proposition

Let C be a linear code. Then $d(C)$ equals the smallest Hamming weight of all nonzero code vectors:
$d(C) = \min \{wt(u) \mid 0 \neq u \in C\}$.

Proof. Since $wt(u) = d(u, 0)$ is the distance between two codewords, we have $wt(u) \geq d(C)$ for all codewords u. It remains to show that there is a codeword with weight equal to $d(C)$. Note that $d(v, w) = wt(v - w)$ for any two vectors v, w. This is because an entry of $v - w$ is nonzero, and hence gets counted in $wt(v - w)$, if and only if v and w differ in

that entry. Choose v and w to be distinct codewords such that $d(v, w) = d(C)$. Then $wt(v - w) = d(C)$, so the minimum weight of the nonzero codewords equals $d(C)$.

To construct a linear $[n, k]$ code, we have to construct a k-dimensional subspace of $\mathbf{F}^n$. The easiest way to do this is to choose k linearly independent vectors and take their span. This can be done by choosing a $k \times n$ **generating matrix** G of rank k, with entries in $\mathbf{F}$. The set of vectors of the form vG, where v runs through all row vectors in $\mathbf{F}^k$, then gives the subspace.

For our purposes, we'll usually take $G = [I_k, P]$, where I_k is the $k \times k$ identity matrix and P is a $k \times (n - k)$ matrix. The rows of G are the basis for a k-dimensional subspace of the space of all vectors of length n. This subspace is our linear code C. In other words, every codeword is uniquely expressible as a linear combination of rows of G. If we use a matrix $G = [I_k, P]$ to construct a code, the first k columns determine the codewords. The remaining $n - k$ columns provide the redundancy.

The code in the second half of Example 1, Section 24.1, has

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The codewords 101010 and 010101 appear as rows in the matrix and the codeword 111111 is the sum of these two rows. This is a $[6, 2]$ code.

The code in Example 2 has

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{matrix}.$$

For example, the codeword 11001001 is the sum mod 2 of the first, second, and fifth rows, and hence is obtained by multiplying $(1, 1, 0, 0, 1, 0, 0)$ times G. This is an $[8, 7]$ code.

In Exercise 4, the matrix G is given in the description of the code. As you can guess from its name, it is a $[7, 4]$ code.

As mentioned previously, we could start with any $k \times n$ matrix of rank k. Its rows would generate an $[n, k]$ code. However, row and column operations can be used to transform the matrix to the form of G we are using, so we usually do not work with the more general situation. A code described by a matrix $G = [I_k, P]$ as before is said to be **systematic**. In this case, the first k bits are the **information symbols** and the last $n - k$ symbols are the **check symbols**.

Suppose we have $G = [I_k, P]$ as the generating matrix for a code C. Let

$$H = [-P^T, I_{n-k}],$$

where P^T is the transpose of P. In Exercise 4 of Section 24.1, this is the matrix that was used to correct errors. For Exercise 2, we have $H = [1, 1, 1, 1, 1, 1, 1, 1]$. Note that in this case a binary string v is a codeword if and only if the number of nonzero bits is even, which is the same as saying that its dot product with H is zero. This can be rewritten as $vH^T = 0$, where H^T is the transpose of H.

More generally, suppose we have a linear code $C \subset \mathbf{F}^n$. A matrix H is called a **parity check matrix** for C if H has the property that a vector $v \in \mathbf{F}^n$ is in C if and only if $vH^T = 0$. We have the following useful result.

Theorem

If $G = [I_k, P]$ is the generating matrix for a code C, then $H = [-P^T, I_{n-k}]$ is a parity check matrix for C.

Proof. Consider the ith row of G, which has the form

$$v_i = (0, \ldots, 1, \ldots, 0, p_{i,1}, \ldots, p_{i,n-k}),$$

where the 1 is in the ith position. This is a vector of the code C. The jth column of H^T is the vector

$$(-p_{1,j}, \ldots, -p_{n-k,j}, 0, \ldots, 1, \ldots, 0),$$

where the 1 is in the $(n-k+j)$th position. To obtain the jth element of $v_i H^T$, take the dot product of these two vectors, which yields

$$1 \cdot (-p_{i,j}) + p_{i,j} \cdot 1 = 0.$$

Therefore, H^T annihilates every row v_i of G. Since every element of C is a sum of rows of G, we find that $vH^T = 0$ for all $v \in C$.

Recall the following fact from linear algebra: The left null space of an $m \times n$ matrix of rank r has dimension $n - r$. Since H^T contains I_{n-k} as a submatrix, it has rank $n - k$. Therefore, its left null space has dimension k. But we have just proved that C is contained in this null space. Since C also has dimension k, it must equal the null space, which is what the theorem claims.

We now have a way of detecting errors: If v is received during a transmission and $vH^T \neq 0$, then there is an error. If $vH^T = 0$, we cannot conclude that there is no error, but we do know that v is a codeword. Since it is more likely that no errors occurred than enough errors occurred to change one codeword into another codeword, the best guess is that an error did not occur.

We can also use a parity check matrix to make the task of decoding easier. First, let's look at an example.

Example

Let C be the binary linear code with generator matrix

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

We are going to make a table of all binary vectors of length 4 according to the following procedure. First, list the four elements of the code in the first row, starting with $(0, 0, 0, 0)$. Then, among the 12 remaining vectors, choose one of smallest weight (there might be several choices). Add this vector to the first row to obtain the second row. From the remaining eight vectors, again choose one with smallest weight and add it to the first row to obtain the third row. Finally, choose a vector with smallest weight from the remaining four vectors, add it to the first row, and obtain the fourth row. We obtain the following:

$$\begin{array}{cccc} (0,0,0,0) & (1,0,1,1) & (0,1,1,0) & (1,1,0,1) \\ (1,0,0,0) & (0,0,1,1) & (1,1,1,0) & (0,1,0,1) \\ (0,1,0,0) & (1,1,1,1) & (0,0,1,0) & (1,0,0,1) \\ (0,0,0,1) & (1,0,1,0) & (0,1,1,1) & (1,1,0,0). \end{array}$$

This can be used as a decoding table. When we receive a vector, find it in the table. Decode by changing the vector to the one at the top of its column. The error that is removed is first element of its row. For example, suppose we receive $(0, 1, 0, 1)$. It is the last element of the second row. Decode it to $(1, 1, 0, 1)$, which means removing the error $(1, 0, 0, 0)$. In this small example, this is not exactly the same as nearest neighbor decoding, since $(0, 0, 1, 0)$ decodes as $(0, 1, 1, 0)$ when it has an equally close neighbor $(0, 0, 0, 0)$. The problem is that the minimum distance of the code is 2, so general error correction is not possible. However, if we use a code that can correct up to t errors, this procedure correctly decodes all vectors that are distance at most t from a codeword.

In a large example, finding the vector in the table can be tedious. In fact, writing the table can be rather difficult (that's why we used such a small example). This is where a parity check matrix H comes to the rescue.

The first vector v in a row is called the **coset leader**. Let r be any vector in the same row as v. Then $r = v + c$ for some codeword c, since this is how the table was constructed. Therefore,

$$rH^T = vH^T + cH^T = vH^T,$$

since $cH^T = 0$ by the definition of a parity check matrix. The vector $S(r) = rH^T$ is called the **syndrome** of r. What we have shown is that two vectors in the same row have the same syndrome. Replace the preceding table with the following much smaller table.

Coset Leader	Syndrome
$(0,\ 0,\ 0,\ 0)$	$(0,\ 0)$
$(1,\ 0,\ 0,\ 0)$	$(1,\ 1)$
$(0,\ 1,\ 0,\ 0)$	$(1,\ 0)$
$(0,\ 0,\ 0,\ 1)$	$(0,\ 1)$

This table may be used for decoding as follows. For a received vector r, calculate its syndrome $S(r) = rH^T$. Find this syndrome on the list and subtract the corresponding coset leader from r. This gives the same decoding as above. For example, if $r = (0, 1, 0, 1)$, then

$$S(r) = (0,\ 1,\ 0,\ 1) \begin{matrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{matrix} = (1,\ 1).$$

This is the syndrome for the second row. Subtract the coset leader $(1, 0, 0, 0)$ from r to obtain the codeword $(1, 1, 0, 1)$.

We now consider the general situation. The method of the example leads us to two definitions.

Definition

Let C be a linear code and let u be an n-dimensional vector. The set $u + C$ given by

$$u + C = \{u + c \mid c \in C\}$$

is called a **coset** of C.

It is easy to see that if $v \in u + C$, then the sets $v + C$ and $u + C$ are the same (Exercise 9).

Definition

A vector having minimum Hamming weight in a coset is called a **coset leader**.

The **syndrome** of a vector u is defined to be $S(u) = uH^T$. The following lemma allows us to determine the cosets easily.

Lemma

Two vectors u and v belong to the same coset if and only if they have the same syndrome.

Proof. Two vectors u and v to belong to the same coset if and only if their difference belongs to the code C; that is, $u - v \in C$. This happens if and only if

$(u-v)H^T = 0$, which is equivalent to
$S(u) = uH^T = vH^T = S(v)$.

Decoding can be achieved by building a syndrome lookup table, which consists of the coset leaders and their corresponding syndromes. With a syndrome lookup table, we can decode with the following steps:

1. For a received vector r, calculate its syndrome $S(r) = rH^T$.
2. Next, find the coset leader with the same syndrome as $S(r)$. Call the coset leader c_0.
3. Decode r as $r - c_0$.

Syndrome decoding requires significantly fewer steps than searching for the nearest codeword to a received vector. However, for large codes it is still too inefficient to be practical. In general, the problem of finding the nearest neighbor in a general linear code is hard; in fact, it is what is known as an NP-complete problem. However, for certain special types of codes, efficient decoding is possible. We treat some examples in the next few sections.

24.4.1 Dual Codes

The vector space $\mathbf{F}^n$ has a dot product, defined in the usual way:

$$(a_1, \ldots, a_n) \cdot (b_0, \ldots, b_n) = a_0 b_0 + \cdots + a_n b_n.$$

For example, if $\mathbf{F} = \mathbf{Z}_2$, then

$$(0, 1, 0, 1, 1, 1) \cdot (0, 1, 0, 1, 1, 1) = 0,$$

so we find the possibly surprising fact that the dot product of a nonzero vector with itself can sometimes be 0, in contrast to the situation with real numbers. Therefore, the dot product does not tell us the length of a vector. But it is still a useful concept.

If C is a linear $[n, k]$ code, define the **dual code**

$$C^{\perp} = \{u \in \mathbf{F}^n \mid u \cdot c = 0 \text{ for all } c \in C\}.$$

Proposition

If C is a linear $[n, k]$ code with generating matrix $G = [I_k, P]$, then $C^{\perp}$ is a linear $[n, n-k]$ code with generating matrix $H = [-P^T, I_{n-k}]$. Moreover, G is a parity check matrix for $C^{\perp}$.

Proof. Since every element of C is a linear combination of the rows of G, a vector u is in $C^{\perp}$ if and only if $uG^T = 0$. This means that $C^{\perp}$ is the left null space of G^T. Also, we see that G is a parity check matrix for $C^{\perp}$. Since G has rank k, so does G^T. The left null space of G^T therefore has dimension $n-k$, so $C^{\perp}$ has dimension $n-k$. Because H is a parity check matrix for C, and the rows of G are in C, we have $GH^T = 0$. Taking the transpose of this relation, and recalling that transpose reverses order ($(AB)^T = B^T A^T$), we find $HG^T = 0$. This means that the rows of H are in the left null space of G^T; therefore, in $C^{\perp}$. Since H has rank $n-k$, the span of its rows has dimension $n-k$, which is the same as the dimension of $C^{\perp}$. It follows that the rows of H span $C^{\perp}$, so H is a generating matrix for $C^{\perp}$.

A code C is called **self-dual** is $C = C^{\perp}$. The Golay code G_{24} of Section 24.6 is an important example of a self-dual code.

Example

Let $C = \{(0,0,0), (1,1,1)\}$ be the binary repetition code. Since $u \cdot (0,0,0) = 0$ for every u, a vector u is in $C^{\perp}$ if and only if $u \cdot (1,1,1) = 0$. This means that $C^{\perp}$

is a parity check code: $(a_1, a_2, a_3) \in C^\perp$ if and only if $a_1 + a_2 + a_3 = 0$.

Example

Let C be the binary code with generating matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

The proposition says that $C^\perp$ has generating matrix

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

This is G with the rows switched, so the rows of G and the rows of H generate the same subspace. Therefore, $C = C^\perp$, which says that C is self-dual.

24.5 Hamming Codes

The Hamming codes are an important class of single error correcting codes that can easily encode and decode. They were originally used in controlling errors in long-distance telephone calls. Binary Hamming codes have the following parameters:

1. Code length: $n = 2^m - 1$
2. Dimension: $k = 2^m - m - 1$
3. Minimum distance: $d = 3$

The easiest way to describe a Hamming code is through its parity check matrix. For a binary Hamming code of length $n = 2^m - 1$, first construct an $m \times n$ matrix whose columns are all nonzero binary m-tuples. For example, for a $[7, 4]$ binary Hamming code we take $m = 3$, so $n = 7$ and $k = 4$, and start with

$$\begin{matrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix}.$$

In order to obtain a parity check matrix for a code in systematic form, we move the appropriate columns to the end so that the matrix ends with the $m \times m$ identity matrix. The order of the other columns is irrelevant. The result is the parity check matrix H for a Hamming $[n, k]$ code. In our example, we move the 4th, 2nd, and 1st columns to the end to obtain

$$H = \begin{matrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{matrix},$$

which is the matrix H from Exercise 3.

We can easily calculate a generator matrix G from the parity check matrix H. Since Hamming codes are single error correcting codes, the syndrome method for decoding can be simplified. In particular, the error vector e is allowed to have weight at most 1, and therefore will be zero or will have all zeros except for a single 1 in the j th position.

The Hamming decoding algorithm, which corrects up to one bit error, is as follows:

1. Compute the syndrome $s = yH^T$ for the received vector y. If $s = 0$, then there are no errors. Return the received vector and exit.

2. Otherwise, determine the position j of the column of H that is the transpose of the syndrome.

3. Change the jth bit in the received word, and output the resulting code.

As long as there is at most one bit error in the received vector, the result will be the codeword that was sent.

Example

The $[15, 11]$ binary Hamming code has parity check matrix

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{matrix}.$$

Assume the received vector is

$$y = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1).$$

The syndrome $s = yH^T$ is calculated to be $s = (1, 1, 1, 1)$. Notice that s is the transpose of the 11th column of H, so we change the 11th bit of y to get the decoded word as

$$(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1).$$

Since the first 11 bits give the information, the original message was

$$(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0).$$

Therefore, we have detected and corrected the error.

24.6 Golay Codes

Two of the most famous binary codes are the Golay codes G_{23} and G_{24}. The [24, 12, 8] extended Golay code G_{24} was used by the *Voyager I* and *Voyager II* spacecrafts during 1979–1981 to provide error correction for transmission back to Earth of color pictures of Jupiter and Saturn. The (nonextended) Golay code G_{23}, which is a [23, 12, 7] code, is closely related to G_{24}. We shall construct G_{24} first, then modify it to obtain G_{23}. There are many other ways to construct the Golay codes. See [MacWilliams-Sloane].

The generating matrix for G_{24} is the 12×24 matrix
$G =$

$$\begin{matrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{matrix}$$

All entries of G are integers mod 2. The first 12 columns of G are the 12×12 identity matrix. The last 11 columns are obtained as follows. The squares mod 11 are 0, 1, 3, 4, 5, 9 (for example, $4^2 \equiv 3$ and $7^2 \equiv 5$). Take the vector
$(x_0, \ldots, x_{10}) = (1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0)$, with a 1 in positions 0, 1, 3, 4, 5, 9. This gives the last 11 entries in the first row of G. The last 11 elements of the other rows, except the last, are obtained by cyclically

permuting the entries in this vector. (Note: The entries are integers mod 2, not mod 11. The squares mod 11 are used only to determine which positions receive a 1.) The 13th column and the 12th row are included because they can be; they increase k and d and help give the code some of its nice properties. The basic properties of G_{24} are given in the following theorem.

Theorem

G_{24} is a self-dual [24, 12, 8] binary code. The weights of all vectors in G_{24} are multiples of 4.

Proof. The rows in G have length 24. Since the 12×12 identity matrix is contained in G, the 12 rows of G are linearly independent. Therefore, G_{24} has dimension 12, so it is a $[24, 12, d]$ code for some d. The main work will be to show that $d = 8$. Along the way, we'll show that G_{24} is self-dual and that the weights of its codewords are $0 \pmod 4$.

Of course, it would be possible to have a computer list all $2^{12} = 4096$ elements of G_{24} and their weights. We would then verify the claims of the theorem. However, we prefer to give a more theoretical proof.

Let r_1 be the first row of G and let $r \neq r_1$ be any of the other first 11 rows. An easy check shows that r_1 and r have exactly four 1s in common, and each has four 1s that are matched with 0s in the other vector. In the sum $r_1 + r$, the four common 1s cancel mod 2, and the remaining four 1s from each row give a total of eight 1s in the sum. Therefore, $r_1 + r$ has weight 8. Also, the dot product $r_1 \cdot r$ receives contributions only from the common 1s, so
$r_1 \cdot r = 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 4 \equiv 0 \pmod 2$
.

Now let u and v be any two distinct rows of G, other than the last row. The first 12 entries and the last 11 entries of v are cyclic permutations of the corresponding parts of u and also of the corresponding parts of the first row. Since a permutation of the entries does not change the weights of vectors or the value of dot products, the preceding calculation of $r_1 + r$ and $r_1 \cdot r$ applies to u and v. Therefore,

1. $wt(u + v) = 8$
2. $u \cdot v \equiv 0 \pmod 2$.

Any easy check shows that (1) and (2) also hold if u or v is the last row of G, so we see that (1) and (2) hold for any two distinct rows u, v of G. Also, each row of G has an even number of 1s, so (2) holds even when $u = v$.

Now let c_1 and c_2 be arbitrary elements of G_{24}. Then c_1 and c_2 are linear combinations of rows of G, so $c_1 \cdot c_2$ is a linear combination of numbers of the form $u \cdot v$ for various rows u and v of G. Each of these dot products is 0 mod 2, so $r_1 \cdot r_2 \equiv 0 \pmod 2$. This implies that $C \subseteq C^\perp$. Since C is a 12-dimensional subspace of 24-dimensional space, $C^\perp$ has dimension $24 - 12 = 12$. Therefore, C and $C^\perp$ have the same dimension, and one is contained in the other. Therefore, $C = C^\perp$, which says that C is self-dual.

Observe that the weight of each row of G is a multiple of 4. The following lemma will be used to show that every element of G_{24} has a weight that is a multiple of 4.

Lemma

Let v_1 and v_2 be binary vectors of the same length. Then

$$wt(v_1 + v_2) = wt(v_1) + wt(v_2) - 2[v_1 \cdot v_2],$$

where the notation $[v_1 \cdot v_2]$ means that the dot product is regarded as a usual integer, not mod 2 (for example, $[(1,0,1,1)\cdot(1,1,1,1)] = 3$, rather than 1).

Proof. The nonzero entries of $v_1 + v_2$ occur when exactly one of the vectors v_1, v_2 has an entry 1 and the other has a 0 as its corresponding entry. When both vectors have a 1, these numbers add to 0 mod 2 in the sum. Note that $wt(v_1) + wt(v_2)$ counts the total number of 1s in v_1 and v_2 and therefore includes these 1s that canceled each other. The contributions to $[v_1 \cdot v_2]$ are caused exactly by these 1s that are common to the two vectors. So there are $[v_1 \cdot v_2]$ entries in v_1 and the same number in v_2 that are included in $wt(v_1) + wt(v_2)$, but do not contribute to $wt(v_1 + v_2)$. Putting everything together yields the equation in the lemma.

We now return to the proof of the theorem. Consider a vector g in G_{24}. It can be written as a sum $g \equiv u_1 + \cdots + u_k \pmod 2$, where $u_1, \ldots, u_k$ are distinct rows of G. We'll prove that $wt(g) \equiv 0 \pmod 4$ by induction on k. Looking at G, we see that the weights of all rows of G are multiples of 4, so the case $k = 1$ is true. Suppose, by induction, that all vectors that can be expressed as a sum of $k-1$ rows of G have weight $\equiv 0 \pmod 4$. In particular, $u = u_1 + \cdots + u_{k-1}$ has weight a multiple of 4. By the lemma,

$$wt(g) = wt(u + u_k) = wt(u) + wt(u_k) - 2[u \cdot u_k] \equiv 0 + 0 - 2[u \cdot u_k] \pmod 4.$$

But $u \cdot u_k \equiv 0 \pmod 2$, as we proved. Therefore, $2[u \cdot u_k] \equiv 0 \pmod 4$. We have proved that $wt(g) \equiv 0 \pmod 4$ whenever g is a sum of k rows. By induction, all sums of rows of G have weight $\equiv 0 \pmod 4$. This proves that all weights of G_{24} are multiples of 4.

Finally, we prove that the minimum weight in G_{24} is 8. This is true for the rows of G, but we also must show it for sums of rows of G. Since the weights of codewords are multiples of 4, we must show that there is no codeword of weight 4, since the weights must then be at least 8. In fact, 8 is then the minimum, because the first row of G, for example, has weight 8.

We need the following lemma.

Lemma

The rows of the 12×12 matrix B formed from the last 12 columns of G are linearly independent mod 2. The rows of the 11×11 matrix A formed from the last 11 elements of the first 11 rows of G are linearly dependent mod 2. The only linear dependence relation is that the sum of all 11 rows of A is 0 mod 2.

Proof. Since G_{24} is self-dual, the dot product of any two rows of G is 0. This means that the matrix product $GG^T = 0$. Since $G = [I|B]$ (that is, I followed by the matrix B), this may be rewritten as

$$I^2 + B\,B^T = 0,$$

which implies that $B^{-1} = B^T$ (we're working mod 2, so the minus signs disappear). This means that B is invertible, so the rows are linearly independent.

The sum of the rows of A is 0 mod 2, so this is a dependence relation. Let $v_1 = (1, \ldots, 1)^T$ be an 11-dimensional column vector. Then $Av_1 = 0$, which is just another way of saying that the sum of the rows is 0. Suppose v_2 is a nonzero 11-dimensional column vector such that $Av_2 = 0$. Extend v_1 and v_2 to 12-dimensional vectors $v_1^{'}$, $v_2^{'}$ by adjoining a 0 at the top of each column vector. Let r_{12} be the bottom row of B. Then

$$Bv_i^{'} = (0, \ldots, 0, r_{12} \cdot v_i^{'})^T.$$

This equation follows from the fact that $Av_i = 0$. Note that multiplying a matrix times a vector consists of taking the dot products of the rows of the matrix with the vector.

Since B is invertible and $v_i^{'} \neq 0$, we have $Bv_i^{'} \neq 0$, so $r_1 \cdot v_i^{'} \neq 0$ Since we are working mod 2, the dot product must equal 1. Therefore,

$$B(v_1^{'} + v_2^{'}) = (0, \ldots, 0, r_1 \cdot v_1^{'} + r_1 \cdot v_2^{'})^T = (0, \ldots, 0, 1 + 1)^T = 0.$$

Since B is invertible, we must have $v_1^{'} + v_2^{'} = 0$, so $v_1^{'} = v_2^{'}$ (we are working mod 2). Ignoring the top entries in $v_1^{'}$ and $v_2^{'}$, we obtain $v_2 = (1, \ldots, 1)$. Therefore, the only nonzero vector in the null space of A is v_1. Since the vectors in the null space of a matrix give the linear dependencies among the rows of the matrix, we conclude that the only dependency among the rows of A is that the sum of the rows is 0. This proves the lemma.

Suppose g is a codeword in G_{24}. If g is, for example, the sum of the second, third, and seventh rows, then g will have 1s in the second, third, and seventh positions, because the first 12 columns of G form an identity matrix. In this way, we see that if g is the sum of k rows of G, then $wt(g) \geq k$. Suppose now that $wt(g) = 4$. Then g is the sum of at most four rows of G. Clearly, g cannot be a single row of G, since each row has weight at least 8. If g is the sum of two rows, we proved that $wt(g)$ is 8. If $g = r_1 + r_2 + r_3$ is the sum of three rows of G, then there are two possibilities.

(1) First, suppose that the last row of G is not one of the rows in the sum. Then three 1s are used from the 13th column, so a 1 appears in the 13th position of g. The 1s from the first 12 positions (one for each of the rows r_1, r_2, r_3) contribute three more 1s to g. Since

$wt(g) = 4$, we have accounted for all four 1s in g. Therefore, the last 11 entries of g are 0. By the preceding lemma, a sum of only three rows of the matrix A cannot be 0. Therefore, this case is impossible.

(2) Second, suppose that the last row of G appears in the sum for g, say $g = r_1 + r_2 + r_3$ with $r_3 =$ the last row of G. Then the last 11 entries of g are formed from the sum of two rows of A (from r_1 and r_2) plus the vector $(1, 1, \ldots, 1)$ from r_3. Recall that the weight of the sum of two distinct rows of G is 8. There is a contribution of 2 to this weight from the first 13 columns. Therefore, looking at the last 11 columns, we see that the sum of two distinct rows of A has weight 6. Adding a vector mod 2 to the vector $(1, 1, \ldots, 1)$ changes all the 1s to 0s and all the 0s to 1s. Therefore, the weight of the last 11 entries of g is 5. Since $wt(g) = 4$, this is impossible, so this case also cannot occur.

Finally, if g is the sum of four rows of G, then the first 12 entries of g have four 1s. Therefore, the last 12 entries of g are all 0. By the lemma, a sum of four rows of B cannot be 0, so we have a contradiction. This completes the proof that there is no codeword of weight 4.

Since the weights are multiples of 4, the smallest possibility for the weight is 8. As we pointed out previously, there are codewords of weight 8, so we have proved that the minimum weight of G_{24} is 8. Therefore, G_{24} is a [24, 12, 8] code, as claimed. This completes the proof of the theorem.

The (nonextended) Golay code G_{23} is obtained by deleting the last entry of each codeword in G_{24}.

Theorem

G_{23} is a linear [23,12,7] code.

Proof. Clearly each codeword has length 23. Also, the set of vectors in G_{23} is easily seen to be closed under addition (if v_1, v_2 are vectors of length 24, then the first 23 entries of $v_1 + v_2$ are computed from the first 23 entries of v_1 and v_2) and G_{23} forms a binary vector space. The generating matrix G' for G_{23} is obtained by removing the last column of the matrix G for G_{24}. Since G' contains the 12×12 identity matrix, the rows of G' are linearly independent, and hence span a 12-dimensional vector space. If g' is a codeword in G_{23}, then g' can be obtained by removing the last entry of some element g of G_{24}. If $g' \neq 0$, then $g \neq 0$, so $wt(g) \geq 8$. Since g' has one entry fewer than g, we have $wt(g') \geq 7$. This completes the proof.

Decoding G_{24}

Suppose a message is encoded using G_{24} and the received message contains at most three errors. In the following, we show a way to correct these errors.

Let G be the 12×24 generating matrix for G_{24}. Write G in the form

$$G = [I, B] = (c_1, \cdots, c_{24}),$$

where I is the 12×12 identity matrix, B consists of the last 12 columns of G, and $c_1, \ldots, c_{24}$ are column vectors. Note that $c_1, \ldots, c_{12}$ are the standard basis elements for 12-dimensional space. Write

$$B^T = (b_1, \ldots, b_{12}),$$

where $b_1, \ldots, b_{12}$ are column vectors. This means that $b_1^T, \ldots, b_{12}^T$ are the rows of B.

Suppose the received message is $r = c + e$, where c is a codeword from G_{24} and

$$e = (e_1, \ldots, e_{24})$$

is the error vector. We assume $wt(e) \leq 3$.

The algorithm is as follows. The justification is given below.

1. Let $s = rG^T$ be the syndrome.
2. Compute the row vectors s, sB, $s + c_j^T$, $13 \leq j \leq 24$, and $sB + b_j^T$, $1 \leq j \leq 12$.
3. If $wt(s) \leq 3$, then the nonzero entries of s correspond to the nonzero entries of e.
4. If $wt(sB) \leq 3$, then there is a nonzero entry in the kth position of sB exactly when the $(k + 12)$th entry of e is nonzero.
5. If $wt(s + c_j^T) \leq 2$ for some j with $13 \leq j \leq 24$, then $e_j = 1$ and the nonzero entries of $s + c_j^T$ are in the positions of the other nonzero entries of the error vector e.
6. If $wt(sB + b_j^T) \leq 2$ for some j with $1 \leq j \leq 12$, then $e_j = 1$. If there is a nonzero entry for this $sB + b_j^T$ in position k (there are at most two such k), then $e_{12+k} = 1$.

Example

The sender starts with the message

$$m = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0).$$

The codeword is computed as

$$mG = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0)$$

and sent to us. Suppose we receive the message as

$$r = (1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0).$$

A calculation shows that

$$s = (0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0)$$

and

$$sB = (1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0).$$

Neither of these has weight at most 3, so we compute $s + c_j^T$, $13 \leq j \leq 24$ and $sB + b_j^T$, $1 \leq j \leq 12$. We find that

$$sB + b_4^T = (0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0).$$

This means that there is an error in position 4 (corresponding to the choice b_4) and in positions 20 (= 12 + 8) and 22 (= 12 + 10) (corresponding to the nonzero entries in positions 8 and 10 of $sB + b_4^T$). We therefore compute

$$\begin{aligned} c = r + &(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0) \\ &= (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0). \end{aligned}$$

Moreover, since G is in systematic form, we recover the original message from the first 12 entries:

$$m = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0).$$

We now justify the algorithm and show that if $wt(e) \leq 3$, then at least one of the preceding cases occurs.

Since G_{24} is self-dual, the dot product of a row of G with any codeword c is 0. This means that $cG^T = 0$. In our case, we have $r = c + e$, so

$$s = rG^T = cG^T + eG^T = eG^T = e_1 c_1^T + \cdots + e_{24} c_{24}^T.$$

This last equality just expresses the fact that the vector $e = (e_1, \ldots, e_{24})$ times the matrix G^T equals e_1 times the first row c_1^T of G^T, plus e_2 times the second row of G^T, etc. Also,

$$sB = eG^T B = e \begin{bmatrix} I \\ B^T \end{bmatrix} B = e \begin{bmatrix} B \\ I \end{bmatrix},$$

since $B^T = B^{-1}$ (proved in the preceding lemma). We have

$$\begin{bmatrix} B \\ I \end{bmatrix} = [B^T, I]^T = (b_1, \ldots, b_{12}, c_1, \ldots, c_{12}).$$

Therefore,

$$sB = e(b_1, \ldots, b_{12}, c_1, \ldots, c_{12})^T = e_1 b_1^T + \ldots + e_{24} c_{12}^T.$$

If $wt(e) \leq 3$, then either $wt((e_1, \ldots, e_{12})) \leq 1$ or $wt((e_{13}, \ldots, e_{24})) \leq 1$, since otherwise there would be too many nonzero entries in e. We therefore consider the following four cases.

1. $wt((e_1, \ldots, e_{12})) = 0$. Then

   $$sB = e_{13} c_1^T + \cdots + e_{24} c_{12}^T = (e_{13}, \ldots, e_{24}).$$

   Therefore, $wt(sB) \leq 3$ and we can determine the errors as in step (4) of the algorithm.

2. $wt((e_1, \ldots, e_{12})) = 1$. Then $e_j = 1$ for exactly one j with $1 \leq j \leq 12$, so

   $$sB = b_j^T + e_{13} c_1^T + \cdots + e_{24} c_{12}^T.$$

   Therefore,

   $$sB + b_j^T = e_{13} c_1^T + \ldots + e_{24} c_{12}^T = (e_{13}, \ldots, e_{24}).$$

   The vector $(e_{13}, \ldots, e_{24})$ has at most two nonzero entries, so we are in step (6) of the algorithm.

   The choice of j is uniquely determined by sB. Suppose $wt(sB + b_k^T) \leq 2$ for some $k \neq j$. Then

   $$wt(b_k^T + b_j^T) = wt(sB + b_k^T + sB + b_j^T)$$
   $$\leq wt(sB + b_k^T) + wt(sB + b_j^T) \leq 2 + 2 = 4$$

   (see Exercise 6). However, we showed in the proof of the theorem about G_{24} that the weight of the sum of any two distinct rows of G has weight 8, from which it follows that the sum of any two distinct rows of B has weight 6. Therefore, $wt(b_k^T + b_j^T) = 6$. This contradiction shows that b_k cannot exist, so b_j is unique.

3. $wt((e_{13}, \ldots, e_{24})) = 0$. In this case,

   $$s = e_1 c_1^T + \cdots + e_{12} c_{12}^T = (e_1, \ldots, e_{12}).$$

   We have $wt(s) \leq 3$, so we are in step (3) of the algorithm.

4. $wt((e_{13}, \ldots, e_{24})) = 1$. In this case, $e_j = 1$ for some j with $13 \leq j \leq 24$. Therefore,

   $$s = e_1 c_1^T + \cdots + e_{12} c_{12}^T + c_j^T,$$

   and we obtain

$$s + c_j^T = e_1 c_1^T + \cdots + e_{12} c_{12}^T = (e_1, \ \ldots, e_{12}).$$

There are at most two nonzero entries in $(e_1, \ \ldots, e_{12})$, so we are in step (5) of the algorithm.

As in (2), the choice of c_j is uniquely determined by s.

In each of these cases, we obtain a vector, let's call it e', with at most three nonzero entries. To correct the errors, we add (or subtract; we are working mod 2) e' to the received vector r to get $c' = r + e'$. How do we know this is the vector that was sent? By the choice of e', we have

$$e' G^T = s,$$

so

$$c' G^T = r G^T + e' G^T = s + s = 0.$$

Since G_{24} is self-dual, G is a parity check matrix for G_{24}. Since $c' G^T = 0$, we conclude that c' is a codeword. We obtained c' by correcting at most three errors in r. Since we assumed there were at most three errors, and since the minimum weight of G_{24} is 8, this must be the correct decoding. So the algorithm actually corrects the errors, as claimed.

The preceding algorithm requires several steps. We need to compute the weights of 26 vectors. Why not just look at the various possibilities for three errors and see which correction yields a codeword? There are $\binom{24}{0} + \binom{24}{1} + \binom{24}{2} + \binom{24}{3} = 2325$ possibilities for the locations of at most three errors, so this could be done on a computer. However, the preceding decoding algorithm is faster.

24.7 Cyclic Codes

Cyclic codes are a very important class of codes. In the next two sections, we'll meet two of the most useful examples of these codes. In this section, we describe the general framework.

A code C is called **cyclic** if

$$(c_1, c_2, \ldots, c_n) \in C \text{ implies } (c_n, c_1, c_2, \ldots, c_{n-1}) \in C.$$

For example, if $(1, 1, 0, 1)$ is in a cyclic code, then so is $(1, 1, 1, 0)$. Applying the definition two more times, we see that $(0, 1, 1, 1)$ and $(1, 0, 1, 1)$ are also codewords, so all cyclic permutations of the codeword are codewords. This might seem to be a strange condition for a code to satisfy. After all, it would seem to be rather irrelevant that, for a given codeword, all of its cyclic shifts are still codewords. The point is that cyclic codes have a lot of structure, which makes them easier to study. In the case of BCH codes (see Section 24.8), this structure yields an efficient decoding algorithm.

Let's start with an example. Consider the binary matrix

$$G = \begin{matrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{matrix}.$$

The rows of G generate a three-dimensional subspace of seven-dimensional binary space. In fact, in this case, the cyclic shifts of the first row give all the nonzero codewords:

$$G = \{(0, 0, 0, 0, 0, 0, 0), (1, 0, 1, 1, 1, 0, 0), (0, 1, 0, 1, 1, 1, 0), (0, 0, 1, 0, 1, 1, 1), \\ (1, 0, 0, 1, 0, 1, 1), (1, 1, 0, 0, 1, 0, 1), (1, 1, 1, 0, 0, 1, 0), (0, 1, 1, 1, 0, 0, 1)\}.$$

Clearly the minimum weight is 4, so we have a cyclic [7, 3, 4] code.

We now show an algebraic way to obtain this code. Let $\mathbf{Z}_2[X]$ denote polynomials in X with coefficients mod 2, and let $\mathbf{Z}_2[X]/(X^7-1)$ denote these polynomials mod (X^7-1). For a detailed description of what this means, see Section 3.11. For the present, it suffices to say that working mod X^7-1 means we are working with polynomials of degree less than 7. Whenever we have a polynomial of degree 7 or higher, we divide by X^7-1 and take the remainder.

Let $g(X) = 1 + X^2 + X^3 + X^4$. Consider all products

$$g(X)f(X) = a_0 + a_1X + \cdots + a_6X^6$$

with $f(X)$ of degree ≤ 2. Write the coefficients of the product as a vector $(a_0, \ \ldots, a_6)$. For example, $g(X) \cdot 1$ yields $(1, 0, 1, 1, 1, 0, 0)$, which is the top row of G. Similarly, $g(X)X$ yields the second row of G and $g(X)X^2$ yields the third row of G. Also, $g(X)(1+X^2)$ yields $(1, 0, 0, 1, 0, 1, 1)$, which is the sum of the first and third rows of G. In this way, we obtain all the codewords of our code.

We obtained this code by considering products $g(X)f(X)$ with $\deg\ (f) \leq 2$. We could also work with $f(X)$ of arbitrary degree and obtain the same code, as long as we work mod (X^7-1). Note that $g(X)(X^3+X^2+1) = X^7 - 1 \pmod 2$. Divide X^3+X^2+1 into $f(X)$:

$$f(X) = (X^3+X^2+1)q(X) + f_1(X),$$

with $\deg\ (f_1) \leq 2$. Then

$$\begin{aligned} g(X)f(X) &= g(X)(X^3+X^2+1)q(X) + g(X)f_1(X) \\ &= (X^7-1)q(X) + g(X)f_1(X) \equiv g(X)f_1(X) \bmod (X^7-1). \end{aligned}$$

Therefore, $g(X)f_1(X)$ gives the same codeword as $g(X)f(X)$, so we may restrict to working with polynomials of degree at most two, as claimed.

Why is the code cyclic? Start with the vector for $g(X)$. The vectors for $g(X)X$ and $g(X)X^2$ are cyclic shifts of the one for $g(X)$ by one place and by two places, respectively. What happens if we multiply by X^3? We obtain a polynomial of degree 7, so we divide by $X^7 - 1$ and take the remainder:

$$g(X)X^3 = X^3 + X^5 + X^6 + X^7 = (X^7 - 1)(1) + (1 + X^3 + X^5 + X^6).$$

The remainder yields the vector $(1, 0, 0, 1, 0, 1, 1)$. This is the cyclic shift by three places of the vector for $g(X)$.

A similar calculation for $j = 4, 5, 6$ shows that the vector for $g(X)X^j$ yields the shift by j places of the vector for $g(X)$. In fact, this is a general phenomenon. If $q(X) = a_0 + a_1X + \cdots + a_6X^6$ is a polynomial, then

$$\begin{aligned} q(X)X &= a_0X + a_1X^2 + \ldots + a_6X^7 \\ &= a_6(X^7 - 1) + a_6 + a_0X + a_1X^2 + \cdots + a_5X^6. \end{aligned}$$

The remainder is $a_6 + a_0X + a_1X^2 + \cdots + a_5X^6$, which corresponds to the vector $(a_6, a_0, \ \ldots, a_5)$. Therefore, multiplying by X and reducing mod $X^7 - 1$ corresponds to a cyclic shift by one place of the corresponding vector. Repeating this j times shows that multiplying by X^j corresponds to shifting by j places.

We now describe the general situation. Let $\mathbf{F}$ be a finite field. For a treatment of finite fields, see Section 3.11. For the present purposes, you may think of $\mathbf{F}$ as being the integers mod p, where p is a prime number, since this is an example of a finite field. For example, you could take $\mathbf{F} = \mathbf{Z}_2 = \{0, 1\}$, the integers mod 2. Let $\mathbf{F}[X]$ denote polynomials in X with coefficients in $\mathbf{F}$. Choose a positive integer n. We'll work in $\mathbf{F}[X]/(X^n - 1)$, which denotes the elements of $\mathbf{F}[X]$ mod $(X^n - 1)$. This means we're working with polynomials of degree less than n. Whenever we encounter a polynomial of degree $\geq n$, we divide by $X^n - 1$ and take the

remainder. Let $g(X)$ be a polynomial in $\mathbf{F}[X]$. Consider the set of polynomials

$$m(X) = g(X)f(X) \bmod (X^n - 1),$$

where $f(X)$ runs through all polynomials in $\mathbf{F}[X]$ (we only need to consider $f(X)$ with degree less than n, since higher-degree polynomials can be reduced mod $X^n - 1$). Write

$$m(X) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}.$$

The coefficients give us the n-dimensional vector $(a_0, \ \ldots, a_{n-1})$. The set of all such coefficients forms a subspace C of n-dimensional space $\mathbf{F}^n$. Then C is a code.

If $m(X) = g(X)f(X) \bmod (X^n - 1)$ is any such polynomial, and $s(X)$ is another polynomial, then $m(X)s(X) = g(X)f(X)s(X) \bmod (X^n - 1)$ is the multiple of $g(X)$ by the polynomial $f(X)s(X)$. Therefore, it yields an element of the code C. In particular, multiplication by X and reducing mod $X^n - 1$ corresponds to a codeword that is a cyclic shift of the original codeword, as above. Therefore, C is cyclic.

The following theorem gives the general description of cyclic codes.

Theorem

Let C be a cyclic code of length n over a finite field $\mathbf{F}$. To each codeword $(a_0, \ \ldots, a_{n-1}) \in C$, associate the polynomial $a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$ in $\mathbf{F}[X]$. Among all the nonzero polynomials obtained from C in this way, let $g(X)$ have the smallest degree. By dividing by its highest coefficient, we may assume that the highest nonzero coefficient of $g(X)$ is 1. The polynomial $g(X)$ is called the **generating polynomial** for C. Then

1. $g(X)$ is uniquely determined by C.
2. $g(X)$ is a divisor of $X^n - 1$.
3. C is exactly the set of coefficients of the polynomials of the form $g(X)f(X)$ with $\deg(f) \leq n - 1 - \deg(g)$.
4. Write $X^n - 1 = g(X)h(X)$. Then $m(X) \in \mathbf{F}[X]/(X^n - 1)$ corresponds to an element of C if and only if $h(X)m(X) \equiv 0 \bmod (X^n - 1)$.

Proof.

1. If $g_1(X)$ is another such polynomial, then $g(X)$ and $g_1(X)$ have the same degree and have highest nonzero coefficient equal to 1. Therefore, $g(X) - g_1(X)$ has lower degree and still corresponds to a codeword, since C is closed under subtraction. Since $g(X)$ had the smallest degree among nonzero polynomials corresponding to codewords, $g(X) - g_1(X)$ must be 0, which means that $g_1(X) = g(X)$. Therefore, $g(X)$ is unique.
2. Divide $g(X)$ into $X^n - 1$:

   $$X^n - 1 = g(X)h(X) + r(X)$$

   for some polynomials $h(X)$ and $r(X)$, with $\deg(r) t \deg(g)$. This means that

   $$-r(X) \equiv g(X)h(X) \bmod (X^n - 1).$$

   As explained previously, multiplying $g(X)$ by powers of X corresponds to cyclic shifts of the codeword associated to $g(X)$. Since C is assumed to be cyclic, the polynomials $g(X)X^j \bmod (X^n - 1)$ for $j = 0, 1, 2, \ldots$ therefore correspond to codewords; call them c_0, c_1, c_2, ... Write $h(X) = b_0 + b_1X + \cdots + b_kX^k$. Then $g(X)h(X)$ corresponds to the linear combination

   $$b_0c_0 + b_1c_1 + \cdots + b_kc_k.$$

   Since each b_i is in $\mathbf{F}$ and each c_i is in C, we have a linear combination of elements of C. But C is a vector subspace of n-dimensional space $\mathbf{F}^n$. Therefore, this linear combination is in C. This means that $r(X)$, which is $g(X)h(X) \mod (X^n - 1)$, corresponds to a codeword. But $\deg(r) t \deg(g)$, which is the minimal degree of a polynomial corresponding to a nonzero codeword in C. Therefore, $r(X) = 0$. Consequently $X^n - 1 = g(X)h(X)$, so $g(X)$ is a divisor of $X^n - 1$.
3. Let $m(X)$ correspond to an element of C. Divide $g(X)$ into $m(X)$:

   $$m(X) = g(X)f(X) + r_1(X),$$

with $\deg\,(r_1(X)) < \deg\,(g(X))$. As before, $g(X)f(X) \mod (X^n - 1)$ corresponds to a codeword. Also, $m(X)$ corresponds to a codeword, by assumption. Therefore, $m(X) - g(X)f(X) \bmod (X^n - 1)$ corresponds to the difference of these codewords, which is a codeword. But this polynomial is just $r_1(X) = r_1(X) \mod (X^n - 1)$. As before, this polynomial has degree less than $\deg\,(g(X))$, so $r_1(X) = 0$. Therefore, $m(X) = g(X)f(X)$. Since $\deg\,(m) \leq n - 1$, we must have $\deg\,((f) \leq n - 1 - \deg\,(g)$. Conversely, as explained in the proof of (2), since C is cyclic, any such polynomial of the form $g(X)f(X)$ yields a codeword. Therefore, these polynomials yield exactly the elements of C.

4. Write $X^n - 1 = g(X)h(X)$, which can be done by (2). Suppose $m(X)$ corresponds to an element of C. Then $m(X) = g(X)f(X)$, by (3), so

$$h(X)m(X) = h(X)g(X)f(X) = (X^n - 1)f(X) \equiv 0 \mod (X^n - 1).$$

Conversely, suppose $m(X)$ is a polynomial such that $h(X)m(X) \equiv 0 \mod (X^n - 1)$. Write $h(X)m(X) = (X^n - 1)q(X) = h(X)g(X)q(X)$, for some polynomial $q(X)$. Dividing by $h(X)$ yields $m(X) = g(X)q(X)$, which is a multiple of $g(X)$, and hence corresponds to a codeword. This completes the proof of the theorem.

Let $g(X) = a_0 + a_1X + \cdots + a_{k-1}X^{k-1} + X^k$ be as in the theorem. By part (3) of the theorem, every element of C corresponds to a polynomial of the form $g(X)f(X)$, with $\deg\,(f(X)) \leq n - 1 - k$. This means that each such $f(X)$ is a linear combination of the monomials $1,\ X,\ X^2,\ \ldots,\ X^{n-1-k}$. It follows that the codewords of C are linear combinations of the codewords corresponding to the polynomials

$$g(X),\ g(X)X,\ g(X)X^2,\ \ldots,\ g(X)X^{n-1-k}.$$

But these are the vectors

$$a_0,\ \ldots,\ a_k,\ 0,\ 0,\ \ldots),\ (0,\ a_0,\ \ldots,\ a_k,\ 0,\ \ldots),\ \ldots,\ (0,\ \ldots,\ 0,\ a_0,\ \ldots,\ a_k).$$

Therefore, a generating matrix for C can be given by

$$G = \begin{matrix} a_0 & a_1 & \cdots & a_k & 0 & 0 & \cdots \\ 0 & a_0 & a_1 & \cdots & a_k & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & a_0 & a_1 & \cdots & a_k \end{matrix}.$$

We can use part (4) of the theorem to obtain a parity check matrix for C. Let $h(X) = b_0 + b_1 X + \cdots + b_l X^l$ be as in the theorem (where $l = n - k$). We'll prove that the $k \times n$ matrix

$$H = \begin{matrix} b_l & b_{l-1} & \cdots & b_0 & 0 & 0 & \cdots \\ 0 & b_l & b_{l-1} & \cdots & b_0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & b_l & b_{l-1} & \cdots & b_0 \end{matrix}$$

is a parity check matrix for C. Note that the order of the coefficients of $h(X)$ is reversed. Recall that H is a parity check matrix for C means that $Hc^T = 0$ if and only if $c \in C$.

Proposition

H is a parity check matrix for C.

Proof. First observe that since $g(X)$ has 1 as its highest nonzero coefficient, and since $g(X)h(X) = X^n - 1$, the highest nonzero coefficient b_l of $h(X)$ must also be 1. Therefore, H is in row echelon form and consequently its rows are linearly independent. Since H has k rows, it has rank k. The right null space of H therefore has dimension $n - k$.

Let $m(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$. We know from part (4) that $(c_0, c_1, \ldots, c_{n-1}) \in C$ if and only if $h(X)m(X) \equiv 0 \bmod (X^n - 1)$.

Choose j with $l \leq j \leq n - 1$ and look at the coefficient of X^j in the product $h(X)m(X)$. It equals

$$b_0 c_j + b_1 c_{j-1} + \cdots + b_{l-1} c_{j-l+1} + b_l c_{j-l}.$$

There is a technical point to mention: Since we are looking at $h(X)m(X) \bmod (X^n - 1)$, we need to worry about a contribution from the term X^{n+j} (since $X^{n+j} \equiv X^n X^j \equiv 1 \cdot X^j$, the monomial X^{n+j} reduces

to X^j). However, the highest-degree term in the product $h(X)m(X)$ before reducing mod $X^n - 1$ is $c_{n-1}X^{l+n-1}$. Since $l \leq j$, we have $l + n - 1tj + n$. Therefore, there is no term with X^{n+j} to worry about.

When we multiply H times $(c_0, c_1, \ldots, c_{n-1})^T$, we obtain a vector whose first entry is

$$b_l c_0 + b_{l-1} c_1 + \cdots + b_0 c_l.$$

More generally, the ith entry (where $1 \leq i \leq k$) is

$$b_l c_{i-1} + b_{l-1} c_i + \cdots + b_0 c_{l+i-1}.$$

This is the coefficient of X^{l+i-1} in the product $h(X)m(X) \bmod (X^n - 1)$.

If $(c_0, c_1, \ldots, c_{n-1})$ is in C, then $h(X)m(X) \equiv 0 \bmod (X^n - 1)$, so all these coefficients are 0. Therefore, H times $(c_0, c_1, \ldots, c_{n-1})^T$ is the 0 vector, so the transposes of the vectors of C are contained in the right null space of H. Since both C and the null space have dimension k, we must have equality. This proves that $c \in C$ if and only if $Hc^T = 0$, which means that H is a parity check matrix for C.

Example

In the example at the beginning of this section, we had $n = 7$ and $g(X) = X^4 + X^3 + X^2 + 1$. We have $g(X)(X^3 + X^2 + 1) = X^7 - 1$, so $h(X) = X^3 + X^2 + 1$. The parity check matrix is

$$H = \begin{matrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}.$$

The parity check matrix gives a way of detecting errors, but correcting errors for general cyclic codes is generally

quite difficult. In the next section, we describe a class of cyclic codes for which a good decoding algorithm exists.

24.8 BCH Codes

BCH codes are a class of cyclic codes. They were discovered around 1959 by R. C. Bose and D. K. Ray-Chaudhuri and independently by A. Hocquenghem. One reason they are important is that there exist good decoding algorithms that correct multiple errors (see, for example, [Gallager] or [Wicker]). BCH codes are used in satellites. The special BCH codes called Reed-Solomon codes (see Section 24.9) have numerous applications.

Before describing BCH codes, we need a fact about finite fields. Let $\mathbf{F}$ be a finite field with q elements. From Section 3.11, we know that $q = p^m$ is a power of a prime number p. Let n be a positive integer not divisible by p. Then it can be proved that there exists a finite field $\mathbf{F}'$ containing $\mathbf{F}$ such that $\mathbf{F}'$ contains a primitive nth root of unity α. This means that $\alpha^n = 1$, but $\alpha^k \neq 1$ for $1 \leq k < n$.

For example, if $\mathbf{F} = \mathbf{Z}_2$, the integers mod 2, and $n = 3$, we may take $\mathbf{F}' = GF(4)$. The element ω in the description of $GF(4)$ in Section 3.11 is a primitive third root of unity. More generally, a primitive nth root of unity exists in a finite field $\mathbf{F}'$ with q' elements if and only if $n | q' - 1$.

The reason we need the auxiliary field $\mathbf{F}'$ is that several of the calculations we perform need to be carried out in this larger field. In the following, when we use an nth root of unity α, we'll implicitly assume that we're calculating in some field $\mathbf{F}'$ that contains α. The results of the calculations, however, will give results about codes over the smaller field $\mathbf{F}$.

The following result, often called the **BCH bound**, gives an estimate for the minimum weight of a cyclic code.

Theorem 24.8

Let C be a cyclic $[n, k, d]$ code over a finite field $\mathbf{F}$, where $\mathbf{F}$ has $q = p^m$ elements. Assume $p \not| n$. Let $g(X)$ be the generating polynomial for C. Let α be a primitive nth root of unity and suppose that for some integers ℓ and δ,

$$g(\alpha^\ell) = g(\alpha^{\ell+1}) = \cdots = g(\alpha^{\ell+\delta}) = 0.$$

Then $d \geq \delta + 2$.

Proof. Suppose $(c_0, c_1, \ldots, c_{n-1}) \in C$ has weight w with $1 \leq w < \delta + 2$. We want to obtain a contradiction. Let $m(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$. We know that $m(X)$ is a multiple of $g(X)$, so

$$m(\alpha^\ell) = m(\alpha^{\ell+1}) = \cdots = m(\alpha^{\ell+\delta}) = 0.$$

Let $c_{i_1}, c_{i_2}, \ldots, c_{i_w}$ be the nonzero coefficients of $m(X)$, so

$$m(X) = c_{i_1} X^{i_1} + c_{i_2} X^{i_2} + \cdots + c_{i_w} X^{i_w}.$$

The fact that $m(\alpha^j) = 0$ for $l \leq j \leq \ell + w - 1$ (note that $w - 1 \leq \delta$) can be rewritten as

$$\begin{pmatrix} \alpha^{\ell i_1} & \cdots & \alpha^{\ell i_w} \\ \alpha^{(\ell+1) i_1} & \cdots & \alpha^{(\ell+1) i_w} \\ \vdots & \ddots & \vdots \\ \alpha^{(\ell+w-1) i_1} & \cdots & \alpha^{(\ell+w-1) i_w} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

We claim that the determinant of the matrix is nonzero. We need the following evaluation of the Vandermonde determinant. The proof can be found in most books on linear algebra.

Proposition

$$\det \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} = \prod_{1 \le itj \le n} (x_j - x_i).$$

(The product is over all pairs of integers (i, j) with $1 \le i < j \le n$.) In particular, if $x_1, \ldots, x_n$ are pairwise distinct, the determinant is nonzero.

In our matrix, we can factor $\alpha^{\ell i_1}$ from the first column, $\alpha^{\ell i_2}$ from the second column, etc., to obtain

$$\det \begin{pmatrix} \alpha^{\ell i_1} & \cdots & \alpha^{\ell i_w} \\ \alpha^{(\ell+1) i_1} & \cdots & \alpha^{(\ell+1) i_w} \\ \vdots & \ddots & \vdots \\ \alpha^{(\ell+w-1) i_1} & \cdots & \alpha^{(\ell+w-1) i_w} \end{pmatrix}$$

$$= \alpha^{\ell i_1 + \cdots + \ell i_w} \det \begin{pmatrix} 1 & \cdots & 1 \\ \alpha^{i_1} & \cdots & \alpha^{i_w} \\ \vdots & \ddots & \vdots \\ \alpha^{(w-1) i_1} & \cdots & \alpha^{(w-1) i_w} \end{pmatrix}.$$

Since $\alpha^{i_1}, \cdots, \alpha^{i_w}$ are pairwise distinct, the determinant is nonzero. Why are these numbers distinct? Suppose $\alpha^{i_j} = \alpha^{i_k}$. We may assume $i_j \le i_k$. We have $0 \le i_j \le i_k < n$. Therefore, $0 \le i_k - i_j < n$. Note that $\alpha^{i_k - i_j} = 1$. Since α is a primitive nth root of unity, $\alpha^i \neq 1$ for $1 \le i < n$. Therefore, $i_k - i_j = 0$, so $i_j = i_k$. This means that the numbers $\alpha^{i_1}, \cdots, \alpha^{i_w}$ are pairwise distinct, as claimed.

Since the determinant is nonzero, the matrix is nonsingular. This implies that $(c_{i_1}, \ldots, c_{i_w}) = 0$, contradicting the fact that these were the nonzero c_i's. Therefore, all nonzero codewords have weight at least $\delta + 2$. This completes the proof of the theorem.

Example

Let $\mathbf{F} = \mathbf{Z}_2$= the integers mod 2, and let $n = 3$. Let $g(X) = X^2 + X + 1$. Then

$$C = \{(0, 0, 0), (1, 1, 1)\},$$

which is a binary repetition code. Let ω be a primitive third root of unity, as in the description of $GF(4)$ in Section 3.11. Then $g(\omega) = g(\omega^2) = 0$. In the theorem, we can therefore take $\ell = 1$ and $\delta = 1$. We find that the minimal weight of C is at least 3. In this case, the bound is sharp, since the minimal weight of C is exactly 3.

Example

Let $\mathbf{F}$ be any finite field and let n be any positive integer. Let $g(X) = X - 1$. Then $g(1) = 0$, so we may take $\ell = 0$ and $\delta = 0$. We conclude that the minimum weight of the code generated by $g(X)$ is at least 2 (actually, the theorem assumes that $p \not| n,$ but this assumption is not needed for this special case where $\ell = \delta = 0$). We have seen this code before. If $(c_0, \ \ldots, c_{n-1})$ is a vector, and $m(X) = c_0 + \cdots + c_{n-1}X^{n-1}$ is the associated polynomial, then $m(X)$ is a multiple of $X - 1$ exactly when $m(1) = 0$. This means that $c_0 + \cdots + c_{n-1} = 0$. So a vector is a codeword if and only if the sum of its entries is 0. When $\mathbf{F} = \mathbf{Z}_2$, this is the parity check code, and for other finite fields it is a generalization of the parity check code. The fact that its minimal weight is 2 is easy to see directly: If a codeword has a nonzero entry, then it must contain another nonzero entry to cancel it and make the sum of the entries be 0. Therefore, each nonzero codeword has at least two nonzero entries, and hence has weight at least 2. The vector $(1, \ -1, 0, \ \ldots)$ is a codeword and has weight 2, so the minimal weight is exactly 2.

Example

Let's return to the example of a binary cyclic code of length 7 from Section 24.7. We have $\mathbf{F} = \mathbf{Z}_2$, and $g(X) = 1 + X^2 + X^3 + X^4$. We can factor $g(X) = (X-1)(X^3 + X + 1)$. Let α be a root of $X^3 + X + 1$. Then α is a primitive seventh root of unity (see Exercise 18), and we are working in $GF(8)$. Since $\mathbf{Z}_2 \subset GF(8)$, we have $2 = 1 + 1 = 0$ and $-1 = 1$. Therefore, $\alpha^3 = \alpha + 1$. Squaring yields $\alpha^6 = \alpha^2 + 2\alpha + 1 = \alpha^2 + 1$. Therefore, $(\alpha^2)^3 + (\alpha^2) + 1 = 0$. This means that $g(\alpha^2) = 0$, so

$$g(1) = g(\alpha) = g(\alpha^2) = 0.$$

In the theorem, we can take $\ell = 0$ and $\delta = 2$. Therefore, the minimal weight in the code is at least 4 (in fact, it is exactly 4).

To define the BCH codes, we need some more notation. We are going to construct codes of length n over a finite field $\mathbf{F}$. Factor $X^n - 1$ into irreducible factors over $\mathbf{F}$:

$$X^n - 1 = f_1(X) f_2(X) \cdots f_r(X),$$

where each $f_i(X)$ is a polynomial with coefficients in $\mathbf{F}$, and each $f_i(X)$ cannot be factored into lower-degree polynomials with coefficients in $\mathbf{F}$. We may assume that the highest nonzero coefficient of each $f_i(X)$ is 1. Let α be a primitive nth root of unity. Then $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{n-1}$ are roots of $X^n - 1$. This means that

$$X^n - 1 = (X-1)(X-\alpha)(X-\alpha^2)\cdots(X-\alpha^{n-1}).$$

Therefore, each $f_i(X)$ is a product of some of these factors $(X - \alpha^j)$, and each α^j is a root of exactly one of the polynomials $f_i(X)$. For each j, let $q_j(X)$ be the polynomial $f_i(X)$ such that $f_i(\alpha^j) = 0$. This gives us polynomials $q_0(X), q_1(X), \ldots q_{n-1}(X)$. Of course, usually these polynomials are not all distinct, since a

polynomial $f_i(X)$ that has two different powers α^j, α^k as roots will serve as both $q_j(X)$ and $q_k(X)$ (see the examples given later in this section).

A **BCH code of designed distance d** is a code with generating polynomial

$$g(X) = \text{least common multiple of } q_{k+1}(X), q_{k+2}(X), \ \ldots, q_{k+d-1}(X)$$

for some integer k.

Theorem

A BCH code of designed distance d has minimum weight greater than or equal to d.

Proof. Since $q_j(X)$ divides $g(X)$ for $k+1 \leq j \leq k+d-1$, and $q_j(\alpha^j) = 0$, we have

$$g(\alpha^{k+1}) = g(\alpha^{k+2}) = \cdots = g(\alpha^{k+d-1}) = 0.$$

The BCH bound (with $\ell = k+1$ and $\delta = d-2$) implies that the code has minimum weight at least $d = \delta + 2$.

Example

Let $\mathbf{F} = \mathbf{Z}_2$, and let $n = 7$. Then

$$X^7 - 1 = (X-1)(X^3 + X^2 + 1)(X^3 + X + 1).$$

Let α be a root of $X^3 + X + 1$. Then α is a primitive 7th root of unity, as in the previous example. Moreover, in that example, we showed that α^2 is also a root of $X^3 + X + 1$. In fact, we actually showed that the square of a root of $X^3 + X + 1$ is also a root, so we have that $\alpha^4 = (\alpha^2)^2$ is also a root of $X^3 + X + 1$. (We could square this again, but $\alpha^8 = \alpha$, so we are back

to where we started.) Therefore, α, α^2, α^4 are the roots of $X^3 + X + 1$, so

$$X^3 + X + 1 = (X - \alpha)(X - \alpha^2)(X - \alpha^4).$$

The remaining powers of α must be roots of $X^3 + X^2 + 1$, so

$$X^3 + X^2 + 1 = (X - \alpha^3)(X - \alpha^5)(X - \alpha^6).$$

Therefore,

$$q_0(X) = X - 1,\ q_1(X) = q_2(X) = q_4(X) = X^3 + X + 1,$$
$$q_3(X) = q_5(X) = q_6(X) = X^3 + X^2 + 1.$$

If we take $k = -1$ and $d = 3$, then

$$g(X) = \text{lcm}\,(q_0(X), q_1(X))$$
$$= (X - 1)(X^3 + X + 1) = X^4 + X^3 + X^2 + 1.$$

We obtain the cyclic $[7, 3, 4]$ code discussed in Section 24.7. The theorem says that the minimum weight is at least 3. In this case, we can do a little better. If we take $k = -1$ and $d = 4$, then we have a generating polynomial $g_1(X)$ with

$$g_1(X) = \text{lcm}(q_0(X), q_1(X), q_2(X)) = g(X).$$

This is because $q_2(X) = q_1(X)$, so the least common multiple doesn't change when $q_2(X)$ is included. The theorem now tells us that the minimum weight of the code is at least 4. As we have seen before, the minimum weight is exactly 4.

Example (continued)

Let's continue with the previous example, but take $k = 0$ and $d = 7$. Then

$$g(X) = \text{lcm}\,(q_1(X),\ \ldots,\ q_6(X)) = (X^3 + X + 1)(X^3 + X^2 + 1)$$
$$= X^6 + X^5 + X^4 + X^3 + X^2 + X + 1.$$

We obtain the repetition code with only two codewords:

$$\{(0,0,0,0,0,0,0),(1,1,1,1,1,1,1)\}.$$

The theorem says that the minimum distance is at least 7. In fact it is exactly 7.

Example

Let $\mathbf{F} = \mathbf{Z}_5 = \{0, 1, 2, 3, 4\}$ = the integers mod 5. Let $n = 4$. Then

$$X^4 - 1 = (X-1)(X-2)(X-3)(X-4)$$

(this is an equality, or congruence if you prefer, in $\mathbf{Z}_5$). Let $\alpha = 2$. We have $2^4 = 1$, but $2^j \neq 1$ for $1 \leq j < 4$. Therefore, 2 is a primitive 4th root of unity in $\mathbf{Z}_5$. We have $2^0 = 1,\ 2^2 = 4,\ 2^3 = 3$ (these are just congruences mod 5). Therefore,

$$q_0(X) = X - 1, \quad q_1(X) = X - 2, \quad q_2(X) = X - 4, \quad q_3(X) = X - 3.$$

In the theorem, let $k = 0$, $d = 3$. Then

$$\begin{aligned} g(X) &= \operatorname{lcm}(q_1(X), q_2(X)) = (X-2)(X-4) \\ &= X^2 - 6X + 8 = X^2 + 4X + 3. \end{aligned}$$

We obtain a cyclic $[4, 2]$ code over $\mathbf{Z}_5$ with generating matrix

$$\begin{pmatrix} 3 & 4 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{pmatrix}.$$

The theorem says that the minimum weight is at least 3. Since the first row of the matrix is a codeword of weight 3, the minimum weight is exactly 3. This code is an example of a Reed-Solomon code, which will be discussed in the next section.

24.8.1 Decoding BCH Codes

One of the reason BCH codes are useful is that there are good decoding algorithms. One of the best known is due

to Berlekamp and Massey (see [Gallager] or [Wicker]). In the following, we won't give the algorithm, but, in order to give the spirit of some of the ideas that are involved, we show a way to correct one error in a BCH code with designed distance $d \geq 3$.

Let C be a BCH code of designed distance $d \geq 3$. Then C is a cyclic code, say of length n, with generating polynomial $g(X)$. There is a primitive nth root of unity α such that

$$g(\alpha^{k+1}) = g(\alpha^{k+2}) = 0$$

for some integer k.

Let

$$H = \begin{pmatrix} 1 & \alpha^{k+1} & \alpha^{2(k+1)} & \dots & \alpha^{(n-1)(k+1)} \\ 1 & \alpha^{k+2} & \alpha^{2(k+2)} & \dots & \alpha^{(n-1)(k+2)} \end{pmatrix}.$$

If $c = (c_0, \ \dots, c_{n-1})$ is a codeword, then the polynomial $m(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$ is a multiple of $g(X)$, so

$$m(\alpha^{k+1}) = m(\alpha^{k+2}) = 0.$$

This may be rewritten in terms of H:

$$cH^T = (c_0, \ c_1, \ \dots, \ c_{n-1}) \begin{pmatrix} 1 & 1 \\ \alpha^{k+1} & \alpha^{k+2} \\ \alpha^{2(k+1)} & \alpha^{2(k+2)} \\ \vdots & \vdots \\ \alpha^{(n-1)(k+1)} & \alpha^{(n-1)(k+2)} \end{pmatrix} = 0.$$

H is not necessarily a parity check matrix for C, since there might be noncodewords that are also in the null space of H. However, as we shall see, H can correct an error.

Suppose the vector $r = c + e$ is received, where c is a codeword and $e = (e_0, \ \dots, e_{n-1})$ is an error vector. We assume that at most one entry of e is nonzero.

Here is the algorithm for correcting one error.

1. Write $rH^T = (s_1, s_2)$.
2. If $s_1 = 0$, there is no error (or there is more than one error), so we're done.
3. If $s_1 \neq 0$, compute s_2/s_1. This will be a power α^{j-1} of α. The error is in the jth position. If we are working over the finite field $\mathbf{Z}_2$, we are done, since then $e_j = 1$. But for other finite fields, there are several choices for the value of e_j.
4. Compute $e_j = s_1/\alpha^{(j-1)(k+1)}$. This is the jth entry of the error vector e. The other entries of e are 0.
5. Subtract the error vector e from the received vector r to obtain the correct codeword c.

Example

Let's look at the BCH code over $\mathbf{Z}_2$ of length 7 and designed distance 7 considered previously. It is the binary repetition code of length 7 and has two codewords: $(0, 0, 0, 0, 0, 0, 0)$, $(1, 1, 1, 1, 1, 1, 1)$. The algorithm corrects one error. Suppose the received vector is $r = (1, 1, 1, 1, 0, 1, 1)$. As before, let α be a root of $X^3 + X + 1$. Then α is a primitive 7th root of unity.

Before proceeding, we need to deduce a few facts about computing with powers of α. We have $\alpha^3 = \alpha + 1$. Multiplying this relation by powers of α yields

$$\begin{gathered}\alpha^4 = \alpha^2 + \alpha, \\ \alpha^5 = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1, \\ \alpha^6 = \alpha^3 + \alpha^2 + \alpha = (\alpha + 1) + \alpha^2 + \alpha = \alpha^2 + 1.\end{gathered}$$

Also, the fact that $\alpha^j = \alpha^{j \,(\mathrm{mod}\, 7)}$ is useful.

We now can compute

$$rH^T = (1,1,1,1,0,1,1)\begin{pmatrix} 1 & 1 \\ \alpha & \alpha^2 \\ \alpha^2 & \alpha^4 \\ \vdots & \vdots \\ \alpha^6 & \alpha^{12} \end{pmatrix}$$
$$= (1+\alpha+\alpha^2+\alpha^3+\alpha^5+\alpha^6, \quad 1+\alpha^2+\alpha^4+\alpha^6+\alpha^{10}+\alpha^{12})$$
$$= (\alpha+\alpha^2, \quad \alpha).$$

The sum in the first entry, for example, can be evaluated as follows:

$$1+\alpha+\alpha^2+\alpha^3+\alpha^5+\alpha^6 = 1+\alpha+\alpha^2+(1+\alpha)+(\alpha^2+\alpha+1)+(\alpha^2+1) = \alpha+\alpha^2.$$

Therefore, $s_1 = \alpha+\alpha^2$ and $s_2 = \alpha$. We need to calculate s_2/s_1. Since $s_1 = \alpha+\alpha^2 = \alpha^4$, we have

$$s_2/s_1 = \alpha/\alpha^4 = \alpha^{-3} = \alpha^4.$$

Therefore, $j-1 = 4$, so the error is in position $j = 5$. The fifth entry of the error vector is $s_1/\alpha^4 = 1$, so the error vector is $(0,0,0,0,1,0,0)$. The corrected message is

$$r - e = (1,1,1,1,1,1,1).$$

Here is why the algorithm works. Since $cH^T = 0$, we have

$$rH^T = cH^T = eH^T = eH^T = (s_1, s_2).$$

If $e = (0, 0, \ldots, e_j, 0, \ldots)$ with $e_j \neq 0$, then the definition of H gives

$$s_1 = e_j\alpha^{(j-1)(k+1)}, \quad s_2 = e_j\alpha^{(j-1)(k+2)}.$$

Therefore, $s_2/s_1 = \alpha^{j-1}$. Also, $s_1/\alpha^{(j-1)(k+1)} = e_j$, as claimed.

24.9 Reed-Solomon Codes

The Reed-Solomon codes, constructed in 1960, are an example of BCH codes. Because they work well for certain types of errors, they have been used in spacecraft communications and in compact discs.

Let $\mathbf{F}$ be a finite field with q elements and let $n = q - 1$. A basic fact from the theory of finite fields is that $\mathbf{F}$ contains a primitive nth root of unity α. Choose d with $1 \leq d < n$ and let

$$g(X) = (X - \alpha)(X - \alpha^2) \cdots (X - \alpha^{d-1}).$$

This is a polynomial with coefficients in $\mathbf{F}$. It generates a BCH code C over $\mathbf{F}$ of length n, called a **Reed-Solomon code**.

Since $g(\alpha) = \cdots = g(\alpha^{d-1}) = 0$, the BCH bound implies that the minimum distance for C is at least d. Since $g(X)$ is a polynomial of degree $d - 1$, it has at most d nonzero coefficients. Therefore, the codeword corresponding to the coefficients of $g(X)$ is a codeword of weight at most d. It follows that the minimum weight for C is exactly d. The dimension of C is $n - \deg(g) = n + 1 - d$. Therefore, a Reed-Solomon code is a cyclic $[n, n + 1 - d, d]$ code.

The codewords in C correspond to the polynomials

$$g(X)f(X) \text{ with } \deg(f) \leq n - d.$$

There are q^{n-d+1} such polynomials $f(X)$ since there are q choices for each of the $n - d + 1$ coefficients of $f(X)$, and thus there are q^{n-d+1} codewords in C. Therefore, a Reed-Solomon code is a MDS code, namely, one that makes the Singleton bound (Section 24.3) an equality.

Example

Let $\mathbf{F} = \mathbf{Z}_7 = \{0, 1, 2, \ldots, 6\}$, the integers mod 7. Then $q = 7$ and $n = q - 1 = 6$. A primitive sixth root of unity α in $\mathbf{F}$ is the same as a primitive root mod 7 (see Section 3.7). We may take $\alpha = 3$. Choose $d = 4$. Then

$$g(X) = (X - 3)(X - 3^2)(X - 3^3) = X^3 + 3X^2 + X + 6.$$

The code has generating matrix

$$G = \begin{matrix} 6 & 1 & 3 & 1 & 0 & 0 \\ 0 & 6 & 1 & 3 & 1 & 0 \\ 0 & 0 & 6 & 1 & 3 & 1 \end{matrix}.$$

There are $7^3 = 343$ codewords in the code, obtained by taking all linear combinations mod 7 of the three rows of G. The minimum weight of the code is 4.

Example

Let $\mathbf{F} = GF(4) = \{0, 1, \omega, \omega^2\}$, which was introduced in Section 3.11. Then $\mathbf{F}$ has 4 elements, $n = q - 1 = 3$, and $\alpha = \omega$. Choose $d = 2$, so

$$g(X) = (X - \omega).$$

The matrix

$$G = \begin{pmatrix} \omega & 1 & 0 \\ 0 & \omega & 1 \end{pmatrix}$$

is a generating matrix for the code. The code contains all 16 linear combinations of the two rows of G, for example,

$$\omega \cdot (\omega, 1, 0) + 1 \cdot (0, \omega, 1) = (\omega^2, 0, 1).$$

The minimum weight of the code is 2.

In many applications, errors are not randomly distributed. Instead, they occur in bursts. For example,

in a CD, a scratch introduces errors in many adjacent bits. A burst of solar energy could have a similar effect on communications from a spacecraft. Reed-Solomon codes are useful in such situations.

For example, suppose we take $\mathbf{F} = GF(2^8)$. The elements of $\mathbf{F}$ are represented as bytes of eight bits each, as in Section 3.11. We have $n = 2^8 - 1 = 255$. Let $d = 33$. The codewords are then vectors consisting of 255 bytes. There are 222 information bytes and 33 check bytes. These codewords are sent as strings of $8 \times 255 = 2040$ binary bits. Disturbances in the transmission will corrupt some of these bits. However, in the case of bursts, these bits will often be in a small region of the transmitted string. If, for example, the corrupted bits all lie within a string of 121 ($= 15 \times 8 + 1$) consecutive bits, there can be errors in at most 16 bytes. Therefore, these errors can be corrected (because $16 < d/2$). On the other hand, if there were 121 bit errors randomly distributed through the string of 2040 bits, numerous bytes would be corrupted, and correct decoding would not be possible. Therefore, the choice of code depends on the type of errors that are expected.

24.10 The McEliece Cryptosystem

In this book, we have mostly described cryptographic systems that are based on number theoretic principles. There are many other cryptosystems that are based on other complex problems. Here we present one based on the difficulty of finding the nearest codeword for a linear binary code.

The idea is simple. Suppose you have a binary string of length 1024 that has 50 errors. There are $\binom{1024}{50} \approx 3 \times 10^{85}$ possible locations for these errors, so an exhaustive search that tries all possibilities is infeasible. Suppose, however, that you have an efficient decoding algorithm that is unknown to anyone else. Then only you can correct these errors and find the corrected string. McEliece showed how to use this to obtain a public key cryptosystem.

Bob chooses G to be the generating matrix for an (n, k) linear error correcting code C with $d(C) = d$. He chooses S to be a $k \times k$ matrix that is invertible mod 2 and lets P be an $n \times n$ permutation matrix, which means that P has exactly one 1 in every row and in every column, with all the other entries being 0. Define

$$G_1 = SGP.$$

The matrix G_1 is the public key for the cryptosystem. Bob keeps S, G, P secret.

In order for Alice to send Bob a message x, she generates a random binary string e of length n that has weight t. She forms the ciphertext by computing

$$y \equiv xG_1 + e \pmod 2.$$

Bob decrypts y as follows:

1. Calculate $y_1 \equiv yP^{-1}$. (Since P is a permutation matrix, $e_1 = eP^{-1}$ is still a binary string of weight t. We have $y_1 \equiv xSG + e_1$.)
2. Apply the error decoder for the code C to y_1 to correct the "error" and obtain the codeword x_1 closest to y_1.
3. Compute x_0 such that $x_0 G \equiv x_1$ (in the examples we have considered, x_0 is simply the first k bits of x_1).
4. Compute $x \equiv x_0 S^{-1}$.

The security of the system lies in the difficulty of decoding y_1 to obtain x_1. There is a little security built into the system by S; however, once a decoding algorithm is known for the code generated by GP, a chosen plaintext attack allows one to solve for the matrix S (as in the Hill cipher).

To make decoding difficult, $d(C)$ should be chosen to be large. McEliece suggested using a $[1024, 512, 101]$ Goppa code. The **Goppa codes** have parameters of the form $n = 2^m$, $d = 2t + 1$, $k = n - mt$. For example, taking $m = 10$ and $t = 50$ yields the $[1024, 524, 101]$ code just mentioned. It can correct up to 50 errors. For given values of m and t, there are in fact many inequivalent Goppa codes with these parameters. We will not discuss these codes here except to mention that they have an efficient decoding algorithm and therefore can be used to correct errors quickly.

Example

Consider the matrix

$$G = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{matrix},$$

which is the generator matrix for the $[7, 4]$ Hamming code. Suppose Alice wishes to send a message

$$m = (1, 0, 1, 1)$$

to Bob. In order to do so, Bob must create an invertible matrix S and a random permutation matrix P that he will keep secret. If Bob chooses

$$S = \begin{matrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$$

and

$$P = \begin{matrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{matrix}.$$

Using these, Bob generates the public encryption matrix

$$G_1 = \begin{matrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{matrix}.$$

In order to encrypt, Alice generates her own random error vector e and calculates the ciphertext $y = xG_1 + e$. In the case of a Hamming code the error vector has weight 1. Suppose Alice chooses

$$e = (0, 1, 0, 0, 0, 0, 0).$$

Then

$$y = (0, 0, 0, 1, 1, 0, 0).$$

Bob decrypts by first calculating

$$y_1 = yP^{-1} = (0, 0, 1, 0, 0, 0, 1).$$

Calculating the syndrome of y_1 by applying the parity check matrix H and changing the corresponding bit yields

$$x_1 = (0, 0, 1, 0, 0, 1, 1).$$

Bob next forms a vector x_0 such that $x_0 G = x_1$, which can be done by extracting the first four components of x_1, that is,

$$x_0 = (0, 0, 1, 0).$$

Bob decrypts by calculating

$$x = x_0 S^{-1} = (1, 0, 1, 1),$$

which is the original plaintext message.

The McEliece system seems to be reasonably secure. For a discussion of its security, see [Chabaud]. A disadvantage of the system compared to RSA, for example, is that the size of the public key G_1 is rather large.

24.11 Other Topics

The field of error correcting codes is a vast subject that is explored by both the mathematical community and the engineering community. In this chapter we have only touched upon a select handful of the concepts of this field. There are many other areas of error correcting codes that we have not discussed.

Perhaps most notable of these is the study of convolutional codes. In this chapter we have entirely focused on block codes, where typically the data are segmented into blocks of a fixed length k and mapped into codewords of a fixed length n. However, in many applications, the data are produced in a continuous fashion, and it is better to map the stream of data into a stream of coded symbols. For example, such codes have the advantage of not requiring the delay needed to observe an entire block of symbols before encoding or decoding. A good analogy is that block codes are the coding theory analogue of block ciphers, while convolutional codes are the analogue of stream ciphers.

Another topic that is very important in the study of error correcting codes is that of efficient decoding. In the case of linear codes, we presented syndrome decoding, which is more efficient than performing a search for the nearest codeword. However, for large linear codes, syndrome decoding is still too inefficient to be practical. When BCH and Reed-Solomon codes were introduced, the decoding schemes that were originally presented were impractical for decoding more than a few errors. Later, Berlekamp and Massey provided an efficient approach to decoding BCH and Reed-Solomon codes. There is still a lot of research being done on this topic. We direct the reader to the books [Lin-Costello], [Wicker], [Gallager], and

[Berlekamp] for further discussion on the subject of decoding.

We have also focused entirely on bit or symbol errors. However, in modern computer networks, the types of errors that occur are not simply bit or symbol errors but also the complete loss of segments of data. For example, on the Internet, data are transferred over the network in chunks called packets. Due to congestion at various locations on the network, such as routers and switches, packets might be dropped and never reach their intended recipient. In this case, the recipient might notify the sender, requesting a packet to be resent. Protocols such as the Transmission Control Protocol (TCP) provide mechanisms for retransmitting lost packets.

When performing cryptography, it is critical to use a combination of many different types of error control techniques to assure reliable delivery of encrypted messages; otherwise, the receiver might not be able to decrypt the messages that were sent.

Finally, we mention that coding theory has strong connections with various problems in mathematics such as finding dense packings of high-dimensional spheres. For more on this, see [Thompson].

24.12 Exercises

1. Two codewords were sent using the Hamming $[7, 4]$ code and were received as 0100111 and 0101010. Each one contains at most one error. Correct the errors. Also, determine the 4-bit messages that were multiplied by the matrix G to obtain the codewords.

2. An ISBN number is incorrectly written as 0-13-116093-8. Show that this is not a correct ISBN number. Find two different valid ISBN numbers such that an error in one digit would give this number. This shows that ISBN cannot correct errors.

3. The following is a parity check matrix for a binary $[n, k]$ code C:

$$\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{matrix}.$$

    1. Find n and k.

    2. Find the generator matrix for C.

    3. List the codewords in C.

    4. What is the code rate for C?

4. Let $C = \{(0, 0, 0), (1, 1, 1)\}$ be a binary repetition code.

    1. Find a parity check matrix for C.

    2. List the cosets and coset leaders for C.

    3. Find the syndrome for each coset.

    4. Suppose you receive the message $(1, 1, 0)$. Use the syndrome decoding method to decode it.

5. Let C be the binary code
$\{(0, 0, 1), (1, 1, 1), (1, 0, 0), (0, 1, 0)\}$.

    1. Show that C is not linear.

    2. What is $d(C)$? (Since C is not linear, this cannot be found by calculating the minimum weight.)

    3. Show that C satisfies the Singleton bound with equality.

6. Show that the weight function (on $\mathbf{F}^n$) satisfies the triangle inequality: $wt(u+v) \leq wt(u) + wt(v)$.

7. Show that $A_q(n, n) = q$, where $A_q(n, d)$ is the function defined in Section 24.3.

8. Let C be the repetition code of length n. Show that $C^\perp$ is the parity check code of length n. (This is true for arbitrary $\mathbf{F}$.)

9. Let C be a linear code and let $u + C$ and $v + C$ be cosets of C. Show that $u + C = v + C$ if and only if $u - v \in C$. (Hint: To show $u + C = v + C$, it suffices to show that $u + c \in v + C$ for every $c \in C$, and that $v + c \in u + C$ for every $c \in C$. To show the opposite implication, use the fact that $u \in u + C$.)

10. Show that if C is a self-dual $[n, k, d]$ code, then n must be even.

11. Show that $g(X) = 1 + X + X^2 + \cdots + X^{n-1}$ is the generating polynomial for the $[n, 1]$ repetition code. (This is true for arbitrary $\mathbf{F}$.)

12. Let $g(X) = 1 + X + X^3$ be a polynomial with coefficients in $\mathbf{Z}_2$.

    1. Show that $g(X)$ is a factor of $X^7 - 1$ in $\mathbf{Z}_2[X]$.

    2. The polynomial $g(X)$ is the generating polynomial for a cyclic code $[7, 4]$ code C. Find the generating matrix for C.

    3. Find a parity check matrix H for C.

    4. Show that $G'H^T = 0$, where

    $$G' = \begin{matrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{matrix}.$$

    5. Show that the rows of G' generate C.

    6. Show that a permutation of the columns of G' gives the generating matrix for the Hamming $[7, 4]$ code, and therefore these two codes are equivalent.

13. Let C be the cyclic binary code of length 4 with generating polynomial $g(X) = X^2 + 1$. Which of the following polynomials correspond to elements of C?

$$f_1(X) = 1 + X + X^3, \quad f_2(X) = 1 + X + X^2 + X^3, \quad f_3(X) = X^2 + X^3$$

14. Let $g(X)$ be the generating polynomial for a cyclic code C of length n, and let $g(X)h(X) = X^n - 1$. Write $h(X) = b_0 + b_1X + \cdots + X^\ell$. Show that the dual code $C^\perp$ is

cyclic with generating polynomial
$\tilde{h}_r(X) = (1/b_0)(1 + b_{\ell-1}X + \cdots + b_1X^{\ell-1} + b_0X^\ell)$. (The factor $1/b_0$ is included to make the highest nonzero coefficient be 1.)

15.
1. Let C be a binary repetition code of odd length n (that is, C contains two vectors, one with all 0s and one with all 1s). Show that C is perfect. (Hint: Show that every vector lies in exactly one of the two spheres of radius $(n-1)/2$.)

2. Use (a) to show that if n is odd then $\sum_{j=0}^{(n-1)/2} \binom{n}{j} = 2^{n-1}$. (This can also be proved by applying the binomial theorem to $(1+1)^n$, and then observing that we're using half of the terms.)

16. Let $2 \le d \le n$ and let $V_q(n, d-1)$ denote the number of points in a Hamming sphere of radius $d-1$. The proof of the Gilbert-Varshamov bound constructs an (n, M, d) code with $M \ge q^n/V_q(n, d-1)$. However, this code is probably not linear. This exercise will construct a linear $[n, k, d]$ code, where k is the smallest integer satisfying $q^k \ge q^{n-1}/V_q(n, d-1)$.

    1. Show that there exists an $[n, 1, d]$ code C_1.

    2. Suppose $q^{j-1} < q^n/V_q(n, d-1)$ and that we have constructed an $[n, j-1, d]$ code C_{j-1} in $\mathbf{F}^n$ (where $\mathbf{F}$ is the finite field with q elements). Show that there is a vector v with $d(v, c) \ge d$ for all $c \in C_{j-1}$.

    3. Let C_j be the subspace spanned by v and C_{j-1}. Show that C_j has dimension j and that every element of C_j can be written in the form $av + c$ with $a \in \mathbf{F}$ and $c \in C_{j-1}$.

    4. Let $av + c$, with $a \ne 0$, be an element of C_j, as in (c). Show that $wt(av + c) = wt(v + a^{-1}c) = d(v, -a^{-1}c) \ge d$.

    5. Show that C_j is an $[n, j, d]$ code. Continuing by induction, we obtain the desired code C_k.

    6. Here is a technical point. We have actually constructed an $[n, k, e]$ code with $e \ge d$. Show that by possibly modifying v in step (b), we may arrange that $d(v, c) = d$ for some $c \in C_{j-1}$, so we obtain an $[n, k, d]$ code.

17. Show that the Golay code G_{23} is perfect.

18. Let α be a root of the polynomial $X^3 + X + 1 \in \mathbf{Z}_2[X]$.

1. Using the fact that $X^3 + X + 1$ divides $X^7 - 1$, show that $\alpha^7 = 1$.

2. Show that $\alpha \neq 1$.

3. Suppose that $\alpha^j = 1$ with $1 \leq j < 7$. Then $\gcd(j, 7) = 1$, so there exist integers a, b with $ja + 7b = 1$. Use this to show that $\alpha^1 = 1$, which is a contradiction. This shows that α is a primitive seventh root of unity.

19. Let C be the binary code of length 7 generated by the polynomial $g(X) = 1 + X^2 + X^3 + X^4$. As in Section 24.8, $g(1) = g(\alpha) = 0$, where α is a root of $X^3 + X + 1$. Suppose the message $(1, 0, 1, 1, 0, 1, 1)$ is received. It has one error. Use the procedure from Section 24.8 to correct the error.

20. Let $C \subset \mathbf{F}^n$ be a cyclic code of length n with generating polynomial $g(X)$. Assume $0 \neq C \neq \mathbf{F}^n$ and $p \not| n$ (as in the theorem on p. 472).

    1. Show that $\deg(g) \geq 1$.

    2. Write $X^n - 1 = g(X)h(X)$. Let α be a primitive nth root of unity. Show that at least one of $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ is a root of $g(X)$. (You may use the fact that $h(X)$ cannot have more than $\deg(h)$ roots.)

    3. Show that $d(C) \geq 2$.

24.13 Computer Problems

1. Three codewords from the Golay code G_{24} are sent and you receive the vectors

$$\begin{array}{cccccccccccccccccccccccc}
(0, & 1, & 0, & 0, & 0, & 0, & 1, & 1, & 0, & 1, & 0, & 1, & 1, & 0, & 1, & 1, & 0, & 1, & 0, & 0, & 0, & 0, & 1, & 1), \\
(0, & 0, & 1, & 0, & 1, & 0, & 1, & 0, & 1, & 0, & 1, & 0, & 0, & 1, & 0, & 0, & 1, & 1, & 1, & 0, & 1, & 1, & 0, & 0), \\
(1, & 1, & 1, & 0, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 0, & 1, & 1, & 1, & 1, & 1, & 1, & 1, & 1).
\end{array}$$

   Correct the errors. (The Golay matrix is stored as *golay* and the matrix B is stored in the downloadable computer files (`bit.ly/2JbcS6p`) as *golaybt*.)

2. An 11-bit message is multiplied by the generating matrix for the Hamming [15, 11] code and the resulting codeword is sent. The vector

$$(0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0)$$

   is received. Assuming there is at most one error, correct it and determine the original 11-bit message. (The parity check matrix for the Hamming [15, 11] code is stored in the downloadable computer files (`bit.ly/2JbcS6p`) as *hammingpc*.)