Chapter 3

Error-Correcting Codes

In the next three chapters, we will present several types of error-correcting codes. A code is a set of messages, called codewords, which can be transmitted electronically between two parties. An error-correcting code is a code for which it is sometimes possible to detect and correct errors that occur during the transmission of codewords. Some applications of error-correcting codes include correction of errors that occur in information transmitted via the Internet, data stored in a computer, and music or video encoded on a compact disc or DVD. Error-correcting codes can also be used to correct errors that occur in information transmitted through space. For example, in Section 3.3, we will briefly discuss a specific error-correcting code that was used in the Mariner 9 space probe when it transmitted photographs back to Earth after entering into an orbit around Mars.

3.1 General Properties

In this chapter, we will look at some types of codes in which the codewords are vectors of a fixed length over $\mathbb{Z}_2$, with entries that we will call bits. We will denote the space of vectors of length $n$ over $\mathbb{Z}_2$ as $\mathbb{Z}_2^n$. Thus, the codes that we will consider in this chapter will be subsets of $\mathbb{Z}_2^n$ for some $n$. A code in $\mathbb{Z}_2^n$ is not required to be a subspace of $\mathbb{Z}_2^n$. If a code is a subspace of $\mathbb{Z}_2^n$, then the code is called a linear code. We will present linear codes beginning in Section 3.6, and then continuing in Chapters 4 and 5.

The way that we will identify in general whether an error has occurred during the transmission of a codeword is to simply check whether the received vector is in the code. As a result, because our goal is to be able to detect and correct errors in received vectors, not all vectors in $\mathbb{Z}_2^n$ can be codewords in a particular code. If a received vector is in the code, then
we will assume that it was the codeword that was sent. If not, then obviously an error occurred in at least one of the bits during transmission, and provided the received vector is of the correct length (i.e., provided no bits were dropped during transmission), we will attempt to correct the received vector to the codeword that was actually sent.

In general, we will use the “nearest neighbor” policy to correct received vectors, meaning we will assume that the fewest possible number of bit errors occurred, and correct a received vector to the codeword from which it differs in the fewest positions. This method of error correction is limited, of course, for there is not always a unique codeword that differs from a received vector in the fewest positions. Also, even when there is a unique codeword that differs from a received vector in the fewest positions, this codeword might not be the vector that was actually sent. However, since the nearest neighbor policy is usually the best approach to take when correcting errors in actual practice, it is the basis from which we will perform our error correction.

**Example 3.1** Consider the code \( C = \{(1010), (1110), (0011)\} \) in \( \mathbb{Z}_2^4 \). Suppose a codeword in \( C \) is transmitted, and we receive the vector \( r_1 = (0110) \). A quick search of \( C \) reveals that \( c = (1110) \) is the codeword from which \( r_1 \) differs in the fewest positions. Thus, we would correct \( r_1 \) to \( c \), and assume that the error in \( r_1 \) was \( r_1 - c = (1000) \).

Now, suppose that a codeword in \( C \) is transmitted, and we receive the vector \( r_2 = (0010) \). Since two of the codewords in \( C \) differ from \( r_2 \) in only one position, we cannot uniquely correct \( r_2 \) using the nearest neighbor policy. So in \( C \), we are not guaranteed to be able to uniquely correct a received vector even if it contains only a single bit error. □

To make the nearest neighbor policy error correction method more precise, we make the following definition. Let \( C \) be a code in \( \mathbb{Z}_2^n \). For codewords \( x, y \in C \), we define the Hamming distance \( d(x, y) \) from \( x \) to \( y \) to be the number of positions in which \( x \) and \( y \) differ. Thus, if \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \), then \( d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \). We will call the smallest Hamming distance between all possible pairs of codewords in a code \( C \) the minimum distance of \( C \), and denote this quantity by \( d(C) \), or just \( d \) if there is no confusion regarding the code to which we are referring. For example, for the code \( C \) in Example 3.1, since the codewords \( (1010) \) and \( (1110) \) differ in only a single position, then \( d = 1 \).

Determining the number of bit errors that are guaranteed to be uniquely correctable in a given code is an important part of coding theory. To do this in general, we consider the following. For \( x \in \mathbb{Z}_2^n \) and positive integer \( r \), let \( S_r(x) = \{ y \in \mathbb{Z}_2^n \mid d(x, y) \leq r \} \), called the ball of radius \( r \) around
3.1. GENERAL PROPERTIES

Suppose \( C \) is a code with minimum distance \( d \), and let \( t \) be the largest integer that satisfies \( t < \frac{d}{2} \). Then \( S_t(x) \cap S_t(y) \) is empty for every pair \( x, y \) of distinct codewords in \( C \). Thus, if \( z \) is a received vector in \( \mathbb{Z}_2^n \) with \( d(u, z) \leq t \) for some codeword \( u \in C \), then \( z \in S_t(u) \), but \( z \notin S_t(v) \) for any other codeword \( v \in C \). That is, if a received vector \( z \) differs from a codeword \( u \) in \( t \) or fewer positions, then every other codeword will differ from \( z \) in more than \( t \) positions. So the nearest neighbor policy will always allow \( t \) or fewer bit errors to be uniquely corrected in the code, and the code is said to be \( t \)-error correcting.

**Example 3.2** Consider the code \( C = \{(00000000), (11100011), (00011111), (11111100)\} \) in \( \mathbb{Z}_2^8 \). It can easily be verified that the minimum distance of \( C \) is \( d = 5 \). Since \( t = 2 \) is the largest integer that satisfies \( t < \frac{d}{2} \), then \( C \) is two-error correcting.

We will now address the problem of determining the number of vectors that are guaranteed to be uniquely correctable to a codeword in a given code. Suppose that \( C \) is a \( t \)-error correcting code in \( \mathbb{Z}_2^n \). Note first that for any \( x \in \mathbb{Z}_2^n \), there will be \( \binom{n}{t} \) vectors in \( \mathbb{Z}_2^n \) that differ from \( x \) in exactly \( k \) positions. Also, any vector in \( \mathbb{Z}_2^n \) that differs from \( x \) in exactly \( k \) positions will be in \( S_t(x) \) provided \( k \leq t \). Thus, the number of vectors in \( S_t(x) \) will be \( \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t} \). To determine the number of vectors in \( \mathbb{Z}_2^n \) that are guaranteed to be uniquely correctable to a codeword in \( C \), we must only count the number of vectors in \( S_t(x) \) as \( x \) ranges through the codewords in \( C \). Since the sets \( S_t(x) \) are pairwise disjoint, the number of vectors in \( \mathbb{Z}_2^n \) that differ from one of the codewords in \( C \) in \( t \) or fewer positions, and which are consequently guaranteed to be uniquely correctable to a codeword in \( C \), is \( |C| \cdot \left( \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t} \right) \). The fact that \( |\mathbb{Z}_2^n| = 2^n \) then yields the following theorem, which gives a bound called the **Hamming bound** on the number of vectors that are guaranteed to be uniquely correctable to a codeword in a \( t \)-error correcting code.

**Theorem 3.1** Suppose that \( C \) is a \( t \)-error correcting code in \( \mathbb{Z}_2^n \). Then

\[
|C| \cdot \left( \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t} \right) \leq 2^n.
\]

A code \( C \) in \( \mathbb{Z}_2^n \) is said to be **perfect** if every vector in \( \mathbb{Z}_2^n \) is guaranteed to be uniquely correctable to a codeword in \( C \). More precisely, a code \( C \) in \( \mathbb{Z}_2^n \) is perfect if the inequality in Theorem 3.1 is an equality. For the code \( C \) in Example 3.2, the factors in this inequality are \( |C| = 4 \), \( \binom{8}{0} + \binom{8}{1} + \binom{8}{2} = 37 \), and \( 2^8 = 256 \). Thus, 108 of the vectors in \( \mathbb{Z}_2^8 \) are not guaranteed to be uniquely correctable to a codeword in this code (although some of these

---

\(^1\)This notation \( \binom{n}{k} \) represents the usual combinatorial value of \( n \ choose \ k \), the number of different ways to choose \( k \) objects from a collection of \( n \) objects.
vectors may still be closest to a unique codeword). So we see that this code is far from perfect. In Section 3.6, we will present a type of code called a Hamming code that is perfect.

In practice, it is usually desirable to construct codes that have a large number of codewords and which are guaranteed to uniquely correct a large number of bit errors. However, the number of bit errors that are guaranteed to be uniquely correctable in a code is obviously related to the number of codewords in the code. Indeed, the problem of constructing a \( t \)-error correcting code \( C \) in \( \mathbb{Z}_2^n \) with \( |C| \) maximized for fixed values of \( n \) and \( t \) has been an important problem of recent mathematical interest. An equivalent problem is to find the maximum number of vectors in \( \mathbb{Z}_2^n \) such that the balls of a fixed radius around the vectors can be arranged in the space without intersecting. This type of problem is called a sphere packing problem.

For the remainder of this chapter and the two subsequent chapters, we will present several methods for constructing various types of codes and correcting errors in these codes. To facilitate this, we will establish a set of parameters that we will use to describe codes. We will describe a code using the parameters \((n, d)\) if the codewords in the code have length \( n \) positions and the code has minimum distance \( d \).

### 3.2 Hadamard Codes

For our first method for constructing codes, we will refer back to block designs, the subject of Chapter 2. The following theorem states that the rows in an incidence matrix for a symmetric block design form the codewords in an error-correcting code.

**Theorem 3.2** Suppose \( A \) is an incidence matrix for a \((v, v, r, r, \lambda)\) block design. Then the rows of \( A \) form a \((v, 2(r - \lambda))\) code with \( v \) codewords.

**Proof.** There are \( v \) positions in each of the \( v \) rows of \( A \). Thus, the rows of \( A \) form a code with \( v \) codewords, each of length \( v \) positions. It remains to be shown only that the minimum distance of this code is \( 2(r - \lambda) \). Consider any pair of rows \( R_1 \) and \( R_2 \) in \( A \). Since every row of \( A \) contains ones in \( r \) positions, and each pair of rows of \( A \) contains ones in common in \( \lambda \) positions, there will be \( r - \lambda \) positions in which \( R_1 \) contains a one and \( R_2 \) contains a zero, and \( r - \lambda \) positions in which \( R_1 \) contains a zero and \( R_2 \) contains a one. This yields \( 2(r - \lambda) \) positions in which \( R_1 \) and \( R_2 \) differ. \( \square \)

**Example 3.3** Theorem 3.2 states that the rows of the incidence matrix \( A \) in Example 2.3 form a \((7, 4)\) code with 7 codewords.

In Theorem 2.5, we showed that a normalized Hadamard matrix of order \( 4k \geq 8 \) can be used to construct an incidence matrix for a