Chapter 2

Error Correcting Codes

The identification number schemes we discussed in the previous chapter give us the ability to determine if an error has been made in recording or transmitting information. However, they are limited in two ways. First, each allows detection of an error in just one digit, expect for some special types of errors, such as interchanging digits. Second, they provide no way to recover the intended information. By making use of more sophisticated ideas and mathematical concepts, we will study methods of encoding and transmitting information that allow us to both detect and correct errors. There are many places that use these so-called error correcting codes, from transmitting photographs from planetary probes to playing of compact discs and dvd movies.

2.1 Basic Notions

To discuss error correcting codes, we need first to set the context and define some terms. We work throughout in binary; that is, we will work over \( \mathbb{Z}_2 \). To simplify notation, we will write the two elements of \( \mathbb{Z}_2 \) as 0 and 1 instead of as \( \overline{0} \) and \( \overline{1} \). If \( n \) is a positive integer, then the set \( \mathbb{Z}_2^n \) is the set of all \( n \)-tuples of \( \mathbb{Z}_2 \)-entries. Elements of \( \mathbb{Z}_2^n \) are called words, or words of length \( n \). A code of length \( n \) is a nonempty subset of \( \mathbb{Z}_2^n \). We will refer to elements of a code as codewords. For convenience we will write elements of \( \mathbb{Z}_2^n \) either with the usual notation, or as a concatenation of digits. For instance, we will write \((0,1,0,1)\) and \(0101\) for the same 4-tuple. We can equip \( \mathbb{Z}_2^n \) with an operation of addition by using point-wise addition. That is, we define

\[
(a_1, \ldots, a_n) + (b_1, \ldots, b_n) = (a_1 + b_1, \ldots, a_n + b_n).
\]

Note that, as a consequence of the facts that \( 0 + 0 = 0 = 1 + 1 \) in \( \mathbb{Z}_2 \), we have \( a + a = 0 \) for any \( a \in \mathbb{Z}_2^n \), where \( 0 \) is the vector \((0, \ldots, 0)\) consisting of all zeros.

**Example 2.1.** The set \( \{01, 10, 11\} \) is a code of length 2, and \( \{0000, 1010, 0101, 1111\} \) is a code of length 4.
Let \( w = a_1 \cdots a_n \) be a word of length \( n \). Then the **weight** of \( w \) is the number of digits of \( w \) equal to 1. We denote the weight of \( w \) by \( \text{wt}(w) \). There are some obvious consequences of this definition. First of all, \( \text{wt}(w) = 0 \) if and only if \( w = 0 \). Second, \( \text{wt}(w) \) is a nonnegative integer. A more sophisticated fact about weight is its relation with addition. If \( v, w \in \mathbb{Z}_2^n \), then \( \text{wt}(v + w) \leq \text{wt}(v) + \text{wt}(w) \). To see why this is true, we write \( x_i \) for the \( i \)-th component of a word \( x \). The weight of \( x \) is then given by the equation \( \text{wt}(x) = |\{i : 1 \leq i \leq n, x_i = 1\}| \).

Using this description of weight, we note that \( (v + w)_i = v_i + w_i \). Therefore, if \( (v + w)_i = 1 \), then either \( v_i = 1 \) or \( w_i = 1 \) (but not both). Therefore,

\[
\{i : 1 \leq i \leq n, (v + w)_i = 1\} \subseteq \{i : v_i = 1\} \cup \{i : w_i = 1\}.
\]

Since \( |A \cup B| \leq |A| + |B| \) for any two finite sets \( A, B \), the inclusion above yields \( \text{wt}(v + w) \leq \text{wt}(v) + \text{wt}(w) \), as desired.

From the idea of weight we can define the notion of distance on \( \mathbb{Z}_2^n \). If \( v, w \) are words, then we set the **distance** \( D(v, w) \) between \( v \) and \( w \) to be

\[
D(v, w) = \text{wt}(v + w).
\]

Alternatively, \( D(v, w) \) is equal to the number of positions in which \( v \) and \( w \) differ. The function \( D \) shares the basic properties of distance in Euclidean space \( \mathbb{R}^3 \). More precisely, it satisfies the properties of the following lemma.

**Lemma 2.2.** The distance function \( D \) defined on \( \mathbb{Z}_2^n \times \mathbb{Z}_2^n \) satisfies

1. \( D(v, v) = 0 \) for all \( v \in \mathbb{Z}_2^n \);
2. for any \( v, w \in \mathbb{Z}_2^n \), if \( D(v, w) = 0 \), then \( v = w \);
3. \( D(v, w) = D(w, v) \) for any \( v, w \in \mathbb{Z}_2^n \);
4. triangle inequality: \( D(v, w) \leq D(v, u) + D(u, w) \) for any \( u, v, w \in \mathbb{Z}_2^n \).

**Proof.** Since \( v + v = 0 \), we have \( D(v, v) = \text{wt}(v + v) = \text{wt}(0) = 0 \). This proves (1). We note that \( 0 \) is the only word of weight 0. Thus, if \( D(v, w) = 0 \), then \( \text{wt}(v + w) = 0 \), which forces \( v + w = 0 \). However, adding \( w \) to both sides yields \( v = w \), and this proves (2). The equality \( D(v, w) = D(w, v) \) is obvious since \( v + w = w + v \). Finally, we prove (4), the only non-obvious statement, with a cute argument. Given \( u, v, w \in \mathbb{Z}_2^n \), we have, from the definition and the fact about weight given above,

\[
D(v, w) = \text{wt}(v + w) = \text{wt}((v + u) + (u + w)) \\
\leq \text{wt}(v + u) + \text{wt}(u + w) \\
= D(v, u) + D(u, w).
\]

\( \square \)
To describe the notion of error detection, we first formalize the notion. Let $C$ be a code. If $w$ is a word, to correct, or decode, $w$ means to select the codeword $v \in C$ such that

$$D(v, w) = \min \{D(u, w) : u \in C\}.$$ 

In other words, we decode $w$ by choosing the closest codeword to $w$, under our notion of distance. There may not be a unique closest codeword, however. When this happens we can either randomly select a closest codeword, or do nothing. We refer to this notion of decoding as maximum likelihood detection, or MLD.

**Example 2.3.** Let $C = \{00111, 11100, 10101\}$. If $w = 00101$, then $w$ is distance 1 from 00111 and distance more than 1 from the other two codewords. Thus, we would decode $w$ as 00111. However, if $u = 10111$, then $u$ is distance 1 from both 00111 and from 10101. Thus, either is an appropriate choice to decode $u$.

We now define what it means for a code to be an error correcting code.

**Definition 2.4.** Let $C$ be a code and let $t$ be a positive integer. Then $C$ is a $t$-error correcting code if whenever a word $w$ differs from the nearest codeword $v$ by a distance of at most $t$, then $v$ is the unique closest codeword to $w$.

If a codeword $v$ is transmitted and received as $w$, we can express $w$ as $v + u$, and we say that $u = v + w$ is the error in transmission. As a word, the error $u$ has a certain weight. So $C$ is $t$-error correcting if for every codeword $v$ and every word $u$ whose weight is at most $t$, then $v$ is the unique closest codeword to $v + u$.

If $C$ is a $t$-error correcting code, then we say that $C$ corrects $t$ errors. Thus one way of interpreting the definition is that if $v$ is a codeword, and if $w$ is obtained from $v$ by changing at most $t$ entries of $v$, then $v$ is the unique closest codeword to $w$. Therefore, by MLD decoding, $w$ will be decoded as $v$.

**Example 2.5.** The code $C = \{00111, 11100, 10101\}$ in the previous example corrects no errors. For, the word $u = 10111$ given in that example is a distance 1 from a codeword, but that codeword is not the unique closest codeword to $u$.

To determine for what $t$ a code corrects $t$ errors, we relate error correction to the distance of a code.

**Definition 2.6.** The distance $d$ of a code is defined by $d = \min \{D(u, v) : u, v \in C, u \neq v\}$.

We denote by $\lfloor a \rfloor$ the greatest integer less than or equal to the number $a$.

**Proposition 2.7.** Let $C$ be a code of distance $d$ and set $t = \lfloor (d - 1)/2 \rfloor$. Then $C$ is a $t$-error correcting code but not a $(t + 1)$-error correcting code.
Example 2.8. Let \( D(v, w) \leq t \). We need to prove that \( v \) is the unique closest codeword to \( w \). We do this by proving that \( D(u, w) > t \) for any codeword \( u \neq v \). If not, suppose that \( u \) is a codeword with \( u \neq v \) and \( D(u, w) \leq t \). Then, by the triangle inequality,

\[
D(u, v) \leq D(u, w) + D(w, v) \leq t + t = 2t < d.
\]

This is a contradiction to the definition of \( d \). Thus, \( v \) is indeed the unique closest codeword to \( w \). To finish the proof, we need to prove that \( C \) does not correct \( t+1 \) errors. Since the code has distance \( d \), there are codewords \( u_1, u_2 \) with \( d = D(u_1, u_2) \). By altering appropriately \( t+1 \) components of \( u_1 \), we can produce a word \( w \) with \( D(u_1, w) = t+1 \) and \( D(w, u_2) = d - (t+1) \). We can do this by considering \( u_1 + u_2 \), a vector with \( d \) components equal to 1, and changing \( d - (t+1) \) of these components to 0, thereby obtaining a word \( e \). We then set \( w = u_1 + e \). Given \( w \), we have \( D(u_1, w) = t + 1 \), but since \( (d - 1) < 2t + 2 \), by definition of \( t \). Thus, \( d - (t+1) < t + 2 \), so \( D(w, u_2) = d - (t+1) \leq t + 1 \). Thus, \( u_1 \) is not the unique closest codeword to \( w \), since \( u_2 \) is either equally close or closer to \( w \). Therefore, \( C \) is not a \((t+1)\)-error correcting code.

We need to show that if \( u \) is any word of weight \( \leq t \) and both \( v \) and \( w \) are codewords, then \( D(v, v + u) < D(w, v + u) \). To see this, first observe that \( D(v + u, v) = wt(u) \), so that \( D(w, v + u) + wt(u) = D(w, v + u) + D(v, v + u) \). The triangle inequality gives \( D(w, v + u) + D(v, v + u) \geq D(w, v) \geq d \) (by definition of \( d \)). Moreover, \( d \geq 2t+1 \geq 2wt(u)+1 \) so that \( D(w, v + u) + wt(u) \geq 2wt(u) + 1 \), and \( D(w, v + u) \geq wt(u) + 1 = D(v, v + u) + 1 \) as desired.

Example 2.8. Let \( C = \{0000, 00111, 11100, 11011\} \). The distance of \( C \) is 3, and so \( C \) is a 1-error correcting code.

Example 2.9. Let \( n \) be an odd positive integer, and let \( C = \{0 \cdots 0, 1 \cdots 1\} \) be a code of length \( n \). If \( n = 2t+1 \), then \( C \) is a \( t \)-error correcting code since the distance of \( C \) is \( n \). Thus, by making the length of \( C \) long enough, we can correct any number of errors that we wish. However, note that the fraction of components of a word that can be corrected is \( t/n \), and this is always less than \( 1/2 \).

2.2 Gaussian Elimination

In this section we discuss the idea of Gaussian elimination for matrices with entries in \( \mathbb{Z}_2 \). We do this now precisely because we need to work with matrices with entries in \( \mathbb{Z}_2 \) in order to discuss the Hamming code, our first example of an error correcting code.

In linear algebra, if you are given a system of linear equations, then you can write this system as a single matrix equation \( AX = b \), where \( A \) is the matrix of coefficients, and \( X \) is
the column matrix of variables. For example, the system
\[
2x + 3y - z = 1 \\
x - y + 5z = 2
\]
is equivalent to the matrix equation
\[
\begin{pmatrix}
2 & 3 & -1 \\
1 & -1 & 5
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
2
\end{pmatrix}.
\]

The primary matrix-theoretic method for solving such a system is to perform Gaussian elimination on the augmented matrix, that matrix that adds to the coefficient matrix one column at the right equal to the column on the right side of the equation. Recall Gaussian elimination performs operations on the rows of a matrix in order to replace the matrix by one in which the solution to the system can be found easily. There are three such row operations:

- multiply or divide a row by a nonzero scalar,
- interchange two rows,
- add a multiple of one row to another row.

It is likely that in all your work with matrices, the entries of the matrices were real numbers. However, to perform the row operations, all you need is to be able to add, subtract, multiply, and divide the entries. In many situations, matrices arise whose entries are not real numbers. For coding theory we need to work with matrices whose entries lie in \(\mathbb{Z}_2 = \{0, 1\}\). Within this set we can add, subtract, multiply, and divide just as if we had real numbers. Furthermore, all the theorems of linear algebra have analogues to the setting where entries lie in \(\mathbb{Z}_2\). In fact, we will generalize the idea of linear algebra later on to include many more sets of scalars. Again, all we need is to be able to perform the four arithmetic operations on the scalars, and we need properties analogous to those that hold for real number arithmetic.

Recall that the only symbolic difference between \(\mathbb{Z}_2\) arithmetic and ordinary arithmetic of these symbols is that \(1 + 1 = 0\) in \(\mathbb{Z}_2\). Note that the first of the three row operations listed above is not useful; multiplying a row by 1 does not affect the row, so is an operation that is not needed. Also, the third operation in the case of \(\mathbb{Z}_2\) reduces to adding one row to another.

Before working some examples, we recall what it means for a matrix to be in row reduced echelon form.

**Definition 2.10.** A matrix \(A\) is in row reduced echelon form if

1. the first nonzero entry of any row is 1. This entry is called a leading 1;
2. If a column contains a leading 1, then all other entries of the column are 0;
3. If \( i > j \), and if row \( i \) and row \( j \) each contain a leading 1, then the column containing the leading 1 of row \( i \) is further to the right than the column containing the leading 1 of row \( j \).

To help understand Condition 3 of the definition, the leading 1’s go to the right as you go from top to bottom in the matrix.

We now give several examples of reducing matrices with \( \mathbb{Z}_2 \) entries to echelon form. In each example once we have the matrix in row reduced echelon form, the leading 1’s are marked in boldface. In understanding the computations below, recall that since \(-1 = 1\) in \( \mathbb{Z}_2 \), subtraction and addition are the same operation.

**Example 2.11.** Consider the matrix

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1
\end{pmatrix}
\]

We reduce the matrix with the following steps. You should determine which row operation was done in each step.

\[
\begin{pmatrix}
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1
\end{pmatrix}.
\]

**Example 2.12.** Consider the matrix

\[
\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1
\end{pmatrix}
\]

To reduce this matrix, we can do the following steps.

\[
\begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}.
\]

**Example 2.13.** To reduce the matrix

\[
\begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1
\end{pmatrix}
\]
we can apply the following single row operation.

\[
\begin{bmatrix}
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}.
\]

We now recall why having a matrix in row reduced echelon form will give us the solution to the corresponding system of equations. The row operations on the augmented matrix corresponds to performing various algebraic manipulations to the equations, such as interchanging equations. So, the system of equations corresponding to the reduced matrix is equivalent to the original system; that is, the two systems have exactly the same solutions.

**Example 2.14.** Consider the system of equations

\[
x = 1 \\
x + y = 1 \\
y + z = 1.
\]

This system has augmented matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1
\end{bmatrix},
\]

and reducing this matrix yields

\[
\begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1
\end{bmatrix}.
\]

This new matrix corresponds to the system of equations

\[
x = 1, \\
y = 0, \\
z = 1.
\]

Thus, we have already the solution to the original system.

**Example 2.15.** The augmented matrix

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1
\end{bmatrix}
\]
corresponds to the system of equations

\[
\begin{align*}
  x_1 + x_2 + x_3 &= 0, \\
  x_1 + x_3 &= 1, \\
  x_2 + x_3 + x_4 + x_5 &= 0, \\
  x_2 + x_3 + x_5 &= 1.
\end{align*}
\]

Reducing the matrix yields

\[
\begin{pmatrix}
  1 & 0 & 1 & 0 & 0 & 1 \\
  0 & 1 & 1 & 0 & 1 & 1 \\
  0 & 0 & 0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix},
\]

which corresponds to the system of equations

\[
\begin{align*}
  x_1 + x_3 &= 1, \\
  x_2 + x_3 + x_5 &= 1, \\
  x_4 &= 1.
\end{align*}
\]

We have left the leading ones in boldface in the echelon matrix. These correspond to the variables \(x_1, x_2,\) and \(x_4\). These variables can be solved in terms of the other variables. Thus, we have the full solution

\[
\begin{align*}
  x_1 &= 1 + x_3, \\
  x_2 &= 1 + x_3 + x_5, \\
  x_4 &= 1, \\
  x_3 \text{ and } x_5 \text{ are arbitrary.}
\end{align*}
\]

We can write out all solutions to this system of equations, since each of \(x_3\) and \(x_5\) can take on the two values 0 and 1. This gives us four solutions, which we write as row vectors.

\[
\begin{align*}
  (x_1, x_2, x_3, x_4, x_5) &= (1, 1, 0, 1, 0), & (x_3 = 0, x_5 = 0) \\
  (x_1, x_2, x_3, x_4, x_5) &= (0, 0, 1, 1, 0), & (x_3 = 1, x_5 = 0) \\
  (x_1, x_2, x_3, x_4, x_5) &= (1, 0, 0, 1, 1), & (x_3 = 0, x_5 = 1) \\
  (x_1, x_2, x_3, x_4, x_5) &= (0, 1, 1, 1, 1), & (x_3 = 1, x_5 = 1).
\end{align*}
\]

**Example 2.16.** Let \(H\) be the *Hamming matrix*

\[
H = \begin{pmatrix}
  0 & 0 & 0 & 1 & 1 & 1 & 1 \\
  0 & 1 & 1 & 0 & 0 & 1 & 1 \\
  1 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix},
\]
and consider the homogeneous system of equations $HX = 0$, where $0$ refers to the $3 \times 1$ zero matrix. Also, $X$ is a $7 \times 1$ matrix of seven variables $x_1, \ldots, x_7$. To solve this system we reduce the augmented matrix

$$
\begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{pmatrix},
$$

yielding

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

This matrix corresponds to the system of equations

\begin{align*}
x_1 + x_3 + x_5 + x_7 &= 0, \\
x_2 + x_3 + x_6 + x_7 &= 0, \\
x_4 + x_5 + x_6 + x_7 &= 0.
\end{align*}

Again, we have marked the leading 1’s in boldface, and the corresponding variables can be solved in terms of the others, which can be arbitrary. So, the solution to this system is

\begin{align*}
x_1 &= x_3 + x_5 + x_7, \\
x_2 &= x_3 + x_6 + x_7, \\
x_4 &= x_5 + x_6 + x_7, \\
x_3, x_5, x_6, x_7 &\text{ are arbitrary.}
\end{align*}

Since we have four variables, $x_3, x_5, x_6,$ and $x_7$, that are arbitrary, and since there are two scalars in $\mathbb{Z}_2$, each variable can take two values. Therefore, we have $2^4 = 16$ solutions to this system of equations.

To finish this chapter, we recall a theorem that will help us determine numeric data about error correcting codes. To state the theorem we need to recall some terminology of linear algebra. We will not bother to define the terms here; you should review them in a linear algebra textbook. We will give the definitions later when we discuss vector spaces in a later chapter. The row space of a matrix is the vector space spanned by the rows of the matrix. If the matrix is $n \times m$, then the rows are $m$-tuples, so the row space is a subspace of the space of all $m$-tuples. The column space of a matrix is the space spanned by the columns of the matrix. Again, if the matrix is $n \times m$, then the columns are $n$-tuples, so the column space is a subspace of the space of all $n$-tuples. The dimension of the row space and the dimension of the column space are always equal. This common positive integer is called the rank of a matrix $A$. One benefit to reducing a matrix $A$ to echelon form is that the rows of the reduced matrix that contain a leading 1 form a basis for the row space of $A$. Consequently,
the dimension of the row space is the number of leading 1’s. Thus, an alternative definition of the rank of a matrix is that it is equal to the number of leading 1’s in the row reduced echelon form obtained from the matrix.

The kernel, or nullspace, of a matrix \(A\) is the set of all solutions to the homogeneous equation \(AX = 0\). To help understand this example, consider the Hamming matrix \(H\) of the previous example.

**Example 2.17.** The solution to the homogeneous equation \(HX = 0\) from the previous example is

\[
\begin{align*}
x_1 &= x_3 + x_5 + x_7, \\
x_2 &= x_3 + x_6 + x_7, \\
x_4 &= x_5 + x_6 + x_7, \\
x_3, x_5, x_6, x_7 & \text{ are arbitrary.}
\end{align*}
\]

For each arbitrary variable we can set it equal to 1 and all other arbitrary variables equal to 0. The resulting vector will be a solution to \(HX = 0\). If we do this for each arbitrary variable, we will have a basis for the nullspace. Doing this, we get the four vectors

\[
\begin{pmatrix}
1 \\
1 \\
1 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}, \quad
\begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
0 \\
0 \\
0
\end{pmatrix}, \quad
\begin{pmatrix}
0 \\
1 \\
0 \\
1 \\
0 \\
0 \\
1
\end{pmatrix}, \quad
\begin{pmatrix}
1 \\
1 \\
0 \\
0 \\
1 \\
1 \\
0
\end{pmatrix}
\]

These vectors do form a basis for the nullspace of \(H\) since the general solution of \(HX = 0\) is

\[
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
x_7
\end{pmatrix} = \begin{pmatrix}
x_3 + x_5 + x_7 \\
x_3 + x_6 + x_7 \\
x_3 \\
x_5 + x_6 + x_7 \\
x_5 \\
x_6 \\
x_7
\end{pmatrix} = x_3 \begin{pmatrix}
1 \\
1 \\
1 \\
0 \\
0 \\
0 \\
0
\end{pmatrix} + x_5 \begin{pmatrix}
0 \\
0 \\
1 \\
1 \\
0 \\
0 \\
0
\end{pmatrix} + x_6 \begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
1 \\
0 \\
0
\end{pmatrix} + x_7 \begin{pmatrix}
1 \\
1 \\
0 \\
0 \\
0 \\
0 \\
1
\end{pmatrix}.
\]

From this equation, we see that every solution is a linear combination of the four specific solutions written above, and a little work will show that every solution can be written in a unique way as a linear combination of these vectors. For example, one can check that \((0, 1, 1, 1, 1, 0, 0)\) is a solution to the system \(HX = 0\), and that to write this vector as a linear combination of the four given vectors, we must have \(x_3 = x_5 = 0\) and \(x_6 = x_7 = 0\),
and so

\[
\begin{pmatrix}
0 \\
1 \\
1 \\
1 \\
1 \\
0 \\
0
\end{pmatrix} = \begin{pmatrix}
1 \\
1 \\
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix} + \begin{pmatrix}
1 \\
0 \\
0 \\
1 \\
1 \\
0 \\
0
\end{pmatrix}
\]

is a sum of two of the four given vectors, and can be written in no other way in terms of the four.

This example indicates the following general fact that for a homogeneous system \( AX = 0 \), that the number of variables not corresponding to leading 1’s is equal to the dimension of the nullspace of \( A \). Let us call these variables leading variables. If we reduce \( A \), the leading variables can be solved in terms of the other variables, and these other variables are all arbitrary; we call them free variables. By mimicking the example above, any solution can be written uniquely in terms of a set of solutions, one for each free variable. This set of solutions is a basis for the nullspace of \( A \); therefore, the number of free variables is equal to the dimension of the nullspace. Every variable is then either a leading variable or a free variable. The number of variables is the number of columns of the matrix. This observation leads to the rank-nullity theorem. The nullity of a matrix \( A \) is the dimension of the nullspace of \( A \).

**Theorem 2.18 (Rank-Nullity).** Let \( A \) be an \( n \times m \) matrix. Then \( m \) is equal to the sum of the rank of \( A \) and the nullity of \( A \).

The point of this theorem is that once you know the rank of \( A \), the nullity of \( A \) can be immediately calculated. The number of solutions to \( AX = 0 \) can then be found. In coding theory this will allow us to determine the number of codewords in a given code.

### 2.3 The Hamming Code

The Hamming code, discovered independently by Hamming and Golay, was the first example of an error correcting code. Let

\[
H = \begin{pmatrix}
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}
\]

be the Hamming matrix, described in Example 2.16 above. Note that the columns of this matrix give the base 2 representation of the integers 1-7. The Hamming code \( C \) of length 7 is the nullspace of \( H \). More precisely,

\[
C = \{ v \in K^7 : Hv^T = 0 \} .
\]
Also by Gaussian elimination, we can solve the linear equation $Hx = 0$, and we get the solution
\[
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5 \\
  x_6 \\
  x_7
\end{pmatrix}
= \begin{pmatrix}
  x_3 \\
  1 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0
\end{pmatrix}
+ \begin{pmatrix}
  x_5 \\
  1 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0
\end{pmatrix}
+ \begin{pmatrix}
  x_6 \\
  0 \\
  0 \\
  0 \\
  0 \\
  0 \\
  1
\end{pmatrix}
+ \begin{pmatrix}
  x_7 \\
  1 \\
  0 \\
  0 \\
  0 \\
  0 \\
  1
\end{pmatrix}.
\]

Therefore, $C$ has dimension 4, and the set \{1110000, 1001100, 0101010, 1101001\} forms a basis for $C$. If we write out all 16 codewords in $C$, we will see that the distance of $C$ is 3. Thus, $C$ is a $(7, 4, 3)$-code. It will then correct 1 error.

The code $C$ has a particularly elegant decoding algorithm, which we now describe. Let $e_1, \ldots, e_7$ be the standard basis for $K^7$. We point out a simple fact of matrix multiplication: $He_i^T$ is equal to the $i$-th column of $H$. Moreover, we note that the 7 nonzero vectors in $K^3$ are exactly the 7 columns of $H$.

Suppose that $v$ is a codeword that is transmitted as a word $w \neq v$. Suppose that exactly one error has been made in transmission. Then $w = v + e_i$ for some $i$. However, we do not yet know $i$, so we cannot yet determine $v$ from $w$. However,
\[
Hw^T = H(v + e_i)^T = Hv^T + He_i^T = He_i^T,
\]
and $He_i^T$ is the $i$-th column of $H$, as we pointed out above. Therefore, we can determine $i$ by computing $Hw^T$ and determining which column of $H$ is this.

The Hamming code $C$ has an additional property: every word is within 1 of a codeword. To see this, suppose that $w$ is a word. If $Hw^T = 0$, then $w$ is a codeword. If not, then $Hw^T$ is a nonzero 3-tuple. Therefore, it is equal to a column of $H$; say that $Hw^T$ is equal to the $i$-th column of $H$. Then $Hw^T = He_i^T$, so $H(w^T + e_i^T) = 0$. Therefore, $w + e_i := v \in C$. The word $v$ is then a codeword a distance of 1 from $w$. A code that corrects $t$ errors and for which every word is within $t$ of some codeword is called perfect. Such codes are particularly nice, in part because a decoding procedure will always return a word. Later we will see some important codes that are not perfect.