# INTRODUCTION TO CODING THEORY: BASIC CODES AND SHANNON'S THEOREM

SIDDHARTHA BISWAS

Abstract. Coding theory originated in the late 1940's and took its roots in engineering. However, it has developed and become a part of mathematics, and especially computer science. Codes were initially developed to correct errors on noisy and inaccurate communication channels. In this endeavor, linear codes are very helpful. Linear codes are simple to understand, yet are the most important and effective for practical applications, such as encoding and decoding messages sent over communication channels. More specifically, we shall examine Hamming codes and their properties. Furthermore, we shall discuss the importance of Shannon's Theorem on the existence of good codes.

Contents

## 1. Linear Codes

Linear codes are some of the most basic codes in coding theory, and are very useful for practical applications. Linear codes are used to encode messages that are then sent over noisy communication channels. For convenience, we shall consider binary codes (i.e. digits in the code are 0 or 1). However, properties and theorems of linear codes still hold true for other number bases.

Consider the message $\mathbf{u} = u_1u_2 \dots u_k$, where each $u_i$ is represented by a 0 or 1. We let $\mathbf{x} = x_1x_2 \dots x_n$ be a function of the message $\mathbf{u}$ such that $n \geq k$.

**Definition 1.1.** A *code* is a set $\mathbf{X}$ such that for all $\mathbf{x} \in \mathbf{X}$, $\mathbf{x}$ is a codeword.

**Definition 1.2.** An *error correcting code* is an algorithm for expressing a sequence of numbers such that any errors which are introduced can be detected and corrected (within certain limitations) based on the remaining numbers.

**Definition 1.3.** We call $\mathbf{x} = \mathbf{x}(\mathbf{u})$ a *linear code* if there is a binary matrix $H$ such that

$$H\mathbf{x}^T = 0. \tag{1.4}$$

for all messages $\mathbf{u}$. We call $H$ the *parity check matrix* of this code.

### 1.1. Properties and Definitions for Linear Codes.

Now that a linear code has been defined let's consider some properties of such a code. We shall also define other basic concepts that helps in further understanding this type of codes.

(1) Given $H$, a parity check matrix of the code $\mathbf{x}$, $H\mathbf{x}^T = 0$.
(2) The parity check matrix $H$ is usually an $(n-k) \times n$ matrix of the form $H = [A|I_{n-k}]$, where $I_{n-k}$ is the identity matrix.
(3) There exists a *generator matrix* $G$, usually a $k \times n$ matrix of the form $G = [I_k| - A^T]$, such that

$$\mathbf{x} = \mathbf{u}G. \tag{1.5}$$

Furthermore, $GH^T = 0$ and $HG^T = 0$.
(4) The code $\mathbf{x} = x_1x_2 \dots x_n$ has *length* $n$. Given the initial message $\mathbf{u} = u_1u_2 \dots u_k$, the code $\mathbf{x}$ has *dimension* $k$. We therefore call $\mathbf{x}$ an $[n, k]$ code.
(5) An $[n, k]$ code has *rank* or *efficiency* $R = k/n$.
(6) Given $\mathbf{x}$ and $\mathbf{y}$ are codes, since $H(\mathbf{x}+\mathbf{y})^T = H\mathbf{x}^T + H\mathbf{y}^T$, $\mathbf{x}+\mathbf{y}$ is therefore also a codeword. This property defines the linearity of $\mathbf{x}$.

### 1.2. Encoding.

When encoding a message into a linear code, the codeword consists of two parts. The first $k$ symbols of the codeword represent the message itself: $x_1 = u_1$, $x_2 = u_2$, ..., $x_k = u_k$. The next $(n-k)$ symbols are called check symbols, and are determined by the parity check matrix and the condition of $H\mathbf{x}^T = 0$.

**Example 1.6.** Let $H$ be the parity check matrix, given by

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

Since $H$ is a $3 \times 6$ matrix, the associated code has length 6 and dimension 3: it is a $[6, 3]$ code. Remember that the first 3 symbols of the codeword give the original message itself so,

$$x_1 = u_1, \quad x_2 = u_2, \quad x_3 = u_3.$$

Therefore, after applying the matrix $H$ to the transpose of $\mathbf{x}$ and substituting the relevant $u_i$, we get the relations

$$\begin{aligned} u_1 + u_3 + x_4 &= 0 \\ u_2 + u_3 + x_5 &= 0 \\ u_1 + x_6 &= 0. \end{aligned}$$

Now, given any message $\mathbf{u} = u_1u_2u_3$, we can create a codeword $\mathbf{x}$. For example, let's consider $\mathbf{x} = 010$. Then,

$$x_4 = 0, \quad x_5 = 1, \quad x_6 = 0.$$

Therefore, our new codeword is $\mathbf{x} = 010010$.

1.3. **Decoding.** Now that the message $\mathbf{u}$ has been encoded into $\mathbf{x}$ and sent through the communication channel, the decoding process can begin. However, it is not always true that the receiver has also received $\mathbf{x}$ as the coded message. Due to channel noise, say that the word received is $\mathbf{y}$. We will now see how to account for error.

**Definition 1.7.** The vector $\mathbf{e} = \mathbf{y} - \mathbf{x} = e_1e_2 \dots e_n$ is defined as the *error vector*.

In order to decode the received message $\mathbf{y}$, it is sufficient to know what $\mathbf{e}$ is, for $\mathbf{x} = \mathbf{y} - \mathbf{e}$ and from $\mathbf{x}$ it is simple to decode the original message $\mathbf{u}$. However, the decoder can never be sure of the error vector $\mathbf{e}$. Therefore, he must pick the one that is most likely to occur. Under the assumption that any codeword $\mathbf{x}$ is equally likely to occur, this method minimizes the probability of the decoder making a mistake, and thus is called *maximum likelihood decoding*. To further describe this decoding method, certain definitions are necessary.

**Definition 1.8.** The *Hamming distance* between two vectors $\mathbf{x} = x_1x_2 \dots x_n$ and $\mathbf{y} = y_1y_2 \dots y_n$ is denoted by $dist(x, y)$ and is equal to the number of places where they differ. For example,

$$dist(111001, 101111) = 3, \quad dist(101010, 001001) = 3.$$

**Definition 1.9.** The *Hamming weight* of a vector $\mathbf{x} = x_1x_2 \dots x_n$ is the number of $x_i \neq 0$, and is written as $wt(\mathbf{u})$. For example,

$$wt(111001) = 4, \quad wt(101010) = 3.$$

**Definition 1.10.** Given a code $\mathbf{X}$, the *minimum distance* of the code is the minimum Hamming distance between its codewords. It is also known as simply the *distance* of the code.

From this, it can be deduced that $dist(\mathbf{x}, \mathbf{y}) = wt(\mathbf{x} - \mathbf{y})$ and $min\ dist(\mathbf{x}, \mathbf{y}) = min\ wt(\mathbf{x} - \mathbf{y})$.

**Definition 1.11.** A *binary symmetric channel* is a channel with binary inputs and binary outputs and error probability $p$.

In a binary symmetric channel, given the error vector $\mathbf{e} = e_1e_2 \dots e_n$, the probability that $e_i = 0$ is $1 - p$ whereas the probability that $e_i = 1$ is $p$ where generally, $0 \leq p < 1/2$. In general, the following equation holds true for the probability of the occurrence $\mathbf{e}$, $\Pr(\mathbf{e})$, given an fixed vector $\mathbf{w}$ of weight $a$.

$$\Pr(e = w) = p^a(1-p)^{n-a}. \tag{1.12}$$

Given that $p < 1/2$, it follows that $(1-p) > p$, and

$$(1-p)^n > p(1-p)^{n-1} > p^2(1-p)^{n-2} > ...$$

Therefore, the error vector with the least weight is most likely, so the decoder picks that vector as the error vector. This method is also known as *nearest neighbor decoding*. From here, a brute force plan is required, as each received codeword $\mathbf{y}$ is compared to all $2^k$ possible codewords $\mathbf{x}$ using the chosen error vector. However, this method is near impossible for $k$ large. Therefore, there must be another method that can decode the message faster.

Another method used to decode codes is using a *standard array*. For this method, we must give a definition.

**Definition 1.13.** Given the set $\mathbf{X}$ of all linear codes $\mathbf{x}$, for any vector $\mathbf{a}$, the set

$$\mathbf{a} + \mathbf{X} = \{\mathbf{a} + \mathbf{x} \mid \mathbf{x} \in \mathbf{X}\}$$

is called a *coset* of $\mathbf{x}$.

Given the received codeword $\mathbf{y}$, $\mathbf{y}$ must belong to some coset. Suppose, $\mathbf{y} = \mathbf{a}_i + \mathbf{x}$, then for the sent codeword $\mathbf{x}'$, the error vector would be $\mathbf{e} = \mathbf{y} - \mathbf{x}' = \mathbf{a}_i + \mathbf{x} - \mathbf{x}' = \mathbf{a}_i + x''$, which is contained in the set $\mathbf{a}_i + \mathbf{X}$. Therefore, all the possible error vectors are the vectors in the coset containing $\mathbf{y}$. The decoder's approach now becomes choosing the minimum weight vector $\hat{\mathbf{e}}$ in this coset; this vector is called the *coset leader*. Now, $\mathbf{y}$ can be decoded as $\hat{\mathbf{x}} = \mathbf{y} - \hat{\mathbf{e}}$. We let $\{\mathbf{a}_i\}$ be the set of coset leaders. We find the coset leaders in a *standard array*.

The standard array is a table, with the first row consisting of the possible original messages $\mathbf{u}$, the second row consisting of the codewords $\mathbf{x}$ with the 0 codeword as the first element of the row, and the other rows consists of the other cosets $\mathbf{a}_i + \mathbf{X}$ with coset leader (error vector) as the first element of each row.

$$\begin{array}{lcccccc}
\text{Row 1} & = & u_1 & u_2 & \dots & u_{2^k} \\
\text{Row 2} & = & x_1 & x_2 & \dots & x_{2^k} \\
\text{Row 3} & = & a_1 + x_1 & a_1 + x_2 & \dots & a_1 + x_{2^k} \\
\vdots & & & & & \\
\text{Row 2} & = & a_n + x_1 & a_n + x_2 & \dots & a_n + x_{2^k}
\end{array}$$

From this standard array, given the received codeword $\mathbf{y}$, we can find the corresponding transmitted codeword $\mathbf{x}$ as the second element of the column $\mathbf{y}$ is in, and from that, the original message $\mathbf{u}$ follows. This is true because when creating this array, each code $\mathbf{y}$ is determined by taking possible codes $\mathbf{x}$ and adding the most likely error vector $\mathbf{e}$ to it. Therefore, the second element of each column of codes $\mathbf{y}$ is the original code $\mathbf{x}$ that was initially sent. This method in theory is fairly simple; however, in practice it can be a long process. There are other methods such as the syndrome decoding method, which will be discussed later.

**Example 1.14.** Given the parity check matrix

$$H = \begin{vmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}$$

we can calculate the generator matrix,

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

From this information, along with the equation $\mathbf{x} = \mathbf{u}G$, we can create the standard array:

| 000 | 001 | 010 | 011 | 100 | 110 | 111 | 101 |
|---|---|---|---|---|---|---|---|
| 000000 | 001101 | 010110 | 011011 | 100001 | 110111 | 111010 | 101100 |
| 100000 | 101101 | 110110 | 111011 | 000001 | 010111 | 011010 | 001100 |
| 010000 | 011101 | 000110 | 001011 | 110001 | 100111 | 101010 | 111100 |
| 001000 | 000101 | 011110 | 010011 | 101001 | 111111 | 110010 | 100100 |
| 000100 | 001001 | 010010 | 011111 | 100101 | 110011 | 111110 | 101000 |
| 000010 | 001111 | 010100 | 011001 | 100011 | 110101 | 111000 | 101110 |
| 000001 | 001100 | 010111 | 011010 | 100000 | 110110 | 111011 | 101101 |

Using this array, given any codeword $\mathbf{y}$, the most likely codeword $\mathbf{x}$ can be derived by the equation $\hat{\mathbf{x}} = \mathbf{y} - \hat{\mathbf{e}}$, and since the first element of each row are error vectors, the codeword $\mathbf{x}$ corresponding to $\mathbf{y}$ is the second element of the column that contains $\mathbf{y}$.

So, if $\mathbf{y} = 011001$, then $\mathbf{x} = 011011$, and following, $\mathbf{u} = 011$.

## 2. Hamming Codes

Hamming codes are a type of binary linear codes developed in 1950 by Richard Hamming. Hamming codes are easy to encode and decode, and are $[n, k, d]$ codes, where $d = \textit{minimum distance}$. Hamming codes are useful in detecting and correcting errors, but first we need to establish the definition of a syndrome.

**Definition 2.1.** Given the received codeword $\mathbf{y}$ and parity check matrix $H$, the *syndrome* of $\mathbf{y}$ is

$$S = H\mathbf{y}^T \tag{2.2}$$

The syndrome identifies errors in the received codeword. The value of the syndrome is the position of the code where the error is. With a binary code, this also implies that the error can be easily corrected. The syndrome tells us the symbol of the code which is erroneous. With a binary code, if the current erroneous symbol is 0, we simply switch it to a 1, and vice versa. Then what is left is the sent codeword which is easy to decode. This is the *syndrome decoding method*.

**Definition 2.3.** The code $\eta$ is a *binary Hamming code* if it is of length $n = 2^r - 1$ (where $r$ is an integer greater than 1) and has a $r \times 2^r - 1$ parity check matrix $H$. That makes $\eta$ a $[2^r - 1, 2^r - 1 - r, 3]$ code.

**Definition 2.4.** Two codes are considered *equivalent* if they differ only by the order of their symbols.

$$\begin{vmatrix} 0000 & 0000 \\ 0001 & 0010 \\ 1001 & 1100 \\ 1111 & 1111 \end{vmatrix}$$

These are examples of 4 sets of equivalent codes where each row is a set of equivalent codes.

**Example 2.5.** Let's consider the Hamming code with $r = 3$. Therefore, we have a $[7, 4, 3]$ code. Let the $(3 \times 7)$ parity check matrix be

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

Notice that columns of the matrix are comprised of the integers 1 through 7 in base two. However, this matrix does not look like the usual $H = [A|I_{n-k}]$ form. Since codes are equivalent if the symbols are in different orders, we can reorder the columns of the matrix to give an equivalent codes. Let

$$H' = \begin{vmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

Now we have the familiar $[A|I_{n-k}]$ form. From this we can use the above methods to encode and decode any 4 digit message.

Hamming codes are linear codes with specific properties. Encoding and decoding them follow the same rules that apply for general linear codes. The importance of Hamming codes lie in the fact that they are error correcting codes that can correct one bit errors. Hamming codes are used in places where such errors are common, such as DRAM chips, satellite communication hardware, and other telecommunications.

## 3. Shannon's Theorem

In 1948, Claude Shannon developed a result that has become one of the fundamental theorems of coding theory. The theorem basically states that error free transmission of codes is possible within a maximum rate depending on the noise of the communication channel. First Shannon established the capacity of a given channel, which is the theoretical transfer rate over that channel.

**Definition 3.1.** The *probability of error* or $\Pr_{\mathbf{e}}$ is the probability that given a received codeword, the decoded output of that codeword is in error.

**Definition 3.2.** The *capacity* of a channel given $\Pr_{\mathbf{e}} = p$ is given by

$$C(p) = 1 + p \log_2(p) + (1 - p) \log_2(1 - p). \tag{3.3}$$

This equation is specifically for binary codes.

**Theorem 3.4.** *Take a binary symmetric channel with symbol error probability. Given any $\epsilon > 0$, $R < C(p)$, for $n$ sufficiently large, there is an $[n, k]$ binary code of rate $k/n \geq R$ with $\Pr_{\mathbf{e}} < \epsilon$.*

From this theorem, we can see that if $R < C(p)$ we can find a $[n, k]$ code that will make the probability error arbitrarily small. With this, communication channel noise and distractions become less important and can be disregarded due to Shannon's remarkable discovery.

## References

[1] F.J. MacWilliams and N.J.A Sloane. The Theory of Error-Correcting Codes. North-Holland Publishing Company. 1978.
[2] Vera Pless. Introduction to the Theory of Error-Correcting Codes, Third Edition. John Wiley and Sons, Inc. 1998.
[3] Proof of Shannon's Theorem, refer to Robert Gallager. Information Theory and Reliable Communication. John Wiley and Sons, Inc. 1968.