Introduction to Coding Theory

♣ **Introduction**. Coding theory originated with the advent of computers. Early computers were huge mechanical monsters whose reliability was low compared to the computers of today. Based, as they were, on banks of mechanical relays, if a single relay failed to close the entire calculation was in error. The engineers of the day devised ways to detect faulty relays so that they could be replaced. While R.W. Hamming was working for Bell Labs, out of frustration with the huge task he was working with, the thought occurred that if the machine was capable of knowing it was in error, wasn't it also possible for the machine to correct that error. Setting to work on this problem Hamming devised a way of encoding information so that if an error was detected it could also be corrected. Based in part on this work, Claude Shannon developed the theoretical framework for the science of coding theory.

Since 1950, the year Hamming published his paper, the theory has been developed for such diverse applications as the minimization of noise from compact disc and DVD recordings, the transfer of financial information across telephone lines, data transfer from one computer to another or from memory to the central processor, and information transmitted from a distance source such as a weather or communication satellite or the Voyager spacecraft which sent pictures of Jupiter and Saturn to Earth.

What we have called Coding Theory, should more properly be called the theory of *Error-Correcting Codes,* since there is another aspect of Coding Theory which is older and deals with the creation and decoding of secret messages. This field is called *Cryptography* and we will not be interested in it. Rather, the problem that we wish to address deals with the difficulties inherent with the transmission of messages. The following diagram provides a rough idea of a general information transmission system.

$$\begin{tikzcd}
\boxed{\textbf{SOURCE}} \arrow[r] & \boxed{\begin{array}{c}\textit{Transmitter}\\ (\textit{Encoder})\end{array}} \arrow[r] & \boxed{\begin{array}{c}\textit{Communication}\\ \textit{Channel}\end{array}} \arrow[r] & \boxed{\begin{array}{c}\textit{Receiver}\\ (\textit{Decoder})\end{array}} \arrow[r] & \boxed{\textbf{SINK}} \\
 & & \boxed{\mathcal{NOISE}} \arrow[u] & &
\end{tikzcd}$$

The most important part of the diagram, as far as we are concerned, is the noise, for without it there would be no need for the theory. Our desire is to construct an information system in such a way as to effect:

1. fast encoding of information,
2. easy transmission of encoded messages,
3. fast decoding of received messages,
4. correction of errors introduced in the channel, and
5. maximum transfer of information per unit time.

The primary goal is the fourth of these. The problem is that it is not generally compatible with the fifth, and also may not be especially compatible with the other three. So any solution is necessarily a trade-off among the five objectives. Before going further, let us introduce some fundamental definitions, assumptions, and terminologies.

♠ **Binary Space**. Let $\mathbb{K} = \{0,1\}$ be the boolean algebra and let $\mathbb{K}^n = \mathbb{K} \times \mathbb{K} \times \cdots \times \mathbb{K}$ be the n-dimensional vector space over $\mathbb{K}$. A member of $\mathbb{K}^n$ will be called a *binary word of* *length n*. Thus 10101110 is a binary word of length 8. Define addition and multiplication of the elements of $\mathbb{K}$ as follows:

$$0+0=0,\ 0+1=1+0=1,\ 1+1=0$$
$$0\,.\,0=0,\ 1\,.\,0=0\,.\,1=0,\ 1\,.\,1=1$$

Define addition for the elements of $\mathbb{K}^n$ componentwise, using the addition defined on $\mathbb{K}$ to add each component. For example,

$$v=1011010 \quad \text{and} \quad w=0110111 \quad \text{then} \quad v+w=1101101.$$

Two words v and w in $\mathbb{K}^n$ are said to be *orthogonal* if their dot product (inner product) is zero. For example, if $v=1101010$ and $w=1011101$, , then

$$v\,.\,w=1\,.\,1+1\,.\,0+0\,.\,1+1\,.\,1+0\,.\,1+1\,.\,0+0\,.\,1=1\,.\,1+1\,.\,1=0$$

♠ **Code**. In many cases, the information to be sent is transmitted by a sequence of zeros and ones. We call a 0 or a 1 a *digit*. A *word* is what is transmitted in place of one piece of information in the original message is a sequence of digits. A *binary code* is a set C of binary words. The code consisting all words of length 3 is

$$C=\{000,100,010,001,110,101,011,111\}=\mathbb{K}^3.$$

A code having all its words of the same length is called a *Block code*; this number is called the *length* of the code. The word that belongs to a code is called a *codeword*. We shall denote the number of codewords in a code by $|C|$. If $C\subseteq\mathbb{K}^n$, then $|C|\leq 2^n=|\mathbb{K}^n|$. In all that follows, we assume that all codes are binary block codes, the zero word will be denoted by $theta$. Also we will assign v to a codeword while w will be assigned to a received word. The *error pattern* u is the word $v+w$.

♠ **Hamming Distance and Weight**. The Hamming distance between two words is the number of places in which they differ. So, for example, the words 001110 and 101100 would have a Hamming distance of 2. This Hamming distance is a metric on the vector space $\mathbb{K}^n$, i.e., if d(x,y) denotes the Hamming distance between word v and w, then d satisfies:

(a) $d(v,v)=0$,
(b) $d(v,w)=d(w,v)$, and
(c) $d(v,w)+d(w,u)\geq d(v,u)$.

Since we will only deal with the Hamming distance (there are other metrics used in Coding Theory), we will generally omit the Hamming and talk about the distance between words. We define a *sphere of radius* r about a word $v\in C$, denoted by $S_r(v)$ as

$$S_r(v)=\{x\in C\ :d(x,v)\leq r\}$$

The *minimum distance* or just the *distance* of a code C, denoted by d is the smallest distance between any pair of distinct codewords in C. It is the minimum distance of a code that measures a code's error-correcting capabilities. If the minimum distance of a code C is $d=2t+1$, then C is a t-error-correcting code, since if t or fewer errors are made in a codeword, the resulting word is closer to the original codeword than it is to any other codeword and so can be correctly decoded.

The *weight* of a word v is the number of non-zero components in the word, denoted by $wt(v)$. We have $d(v,w) = wt(v+w)$. Alternatively, the weight is the distance of the word from the zero word θ. The *minimum weight* of a code is the weight of nonzero codeword of smallest weight in the code. So minimum weight is the same as the minimum distance. We shall see that examining the weights of the codewords gives useful information about a particular code.

♠ **Noise**. Undesirable disturbances that may cause the information received to differ from what was transmitted. Noise may be caused by sunspots, lightning, folds in a magnetic tape, meteor showers, competing telephone messages, random radio disturbances, poor hearing, poor speech, or many other things.

♠ **Channel**. The physical medium through which the information is transmitted. Telephone lines and the atmosphere are examples of channels. In practice, the control we have over the noise is the choice of a good channel to use for transmission and the use of various noise filters to combat certain types of interference which may be encountered. A *binary channel* is a channel that uses zeros and ones. We need to make some basic assumptions about the channel.

1. The channel is a binary channel.
2. A binary codeword v of length n is received as a binary word w which may or may not be the same as the transmitted codeword v.
3. There is no difficulty identifying the beginning of the first word transmitted. Thus, if we are using codewords of length 3 and receive 110111010, we know that the words received are, in order, 110, 111, 010. This assumption means that the channel cannot deliver 11011 to the receiver, because a digit has been lost here.
4. The noise is scattered randomly as opposed to being in clumps called *bursts.*That is, the probability of any one digit being affected in transmission is the same as that of any other digit and is not influenced by errors made in neighboring digits. This is not a very realistic assumption for many types of noise such as scratches on compact discs.

In a *perfect,* or noiseless, channel, the digit sent, 0 or 1, is always the digit received. If all channels were perfect, there would be no need for coding theory. But no channel is perfect; every channel is noisy. Some less noisy, or more reliable, than others. A binary channel is *symmetric* if 0 and 1 are transmitted with equal accuracy; that is the probability of receiving the correct digit is independent of which digit, 0 or 1, is being transmitted. The reliability of a binary symmetric channel $(\mathcal{BSC})$ is a real number p, $0 \le p \le 1$, where p is the probability that the digit sent is the digit received.

If p is the probability that the digit received is the same as the digit sent, then $q = 1-p$ is the probability that the digit received is *not* the digit sent. The following diagram may clarify how a $\mathcal{BSC}$ operates:

$$\begin{tikzcd}
0 \arrow[r, "p"] \arrow[dr, "q" near start] & 0 \\
1 \arrow[r, "p"'] \arrow[ur] & 1
\end{tikzcd}$$

In most cases it may be hard to estimate the actual value of p for a given channel. We will call one channel more reliable than other if its reliability is higher. Note that if $p = 1$, then the channel is perfect of no interest to us. Nor is channel with $p = 0$ or $p = 1/2$ of any interest. Any channel with $0 < p < 1/2$ can be easily converted into a channel

$1/2 < p < 1$. Hence we will always assume that we are using a $\mathcal{BSC}$ with probably p satisfying $1/2 < p < 1$.

♣ **Correcting and Detecting Error Patterns**. Suppose that we wished to transmit a message and knew that in the process of transmission there would be some altering of the message, due to weak signals, sporadic electrical bursts and other naturally occurring noise that creeps into the transmission medium. The problem is to insure that the intended message is obtainable from whatever is actually received. One simple approach to this problem is what is called a repeat code. For instance, if we wanted to send the message $\mathcal{BAD\ NEWS}$, we could repeat each letter a certain number of times and send, say,

$$\mathcal{GGGGGOOOOODDDDD} \qquad \mathcal{NNNNNEEEEEWWWWWSSSSS}.$$

Even if a number of these letters got garbled in transmission, the intended message could be recovered from a received message that might look like

$$\mathcal{GRGGGOOOKODCDDD} \qquad \mathcal{MNNNXEEEEEWWWSWSSTST},$$

by a process called *majority decoding*, which in this case would mean that for each block of 5 letters the intended letter is the one which appears most frequently in the block. The problem with this approach is economics, the repeat code is not very efficient. The increased length of the transmitted code, and thus the increased time and energy required to transmit it, is necessary in order to be able to decode the message properly, but how efficiently a coding procedure uses this increase depends upon the coding scheme.

Let $C_1 = \{00, 01, 10, 11\}$. Every received word is a codeword and so C_1 cannot detect any errors. Also C_1 corrects no errors since every received word requires no changes to become a codeword.

Modify C_1 into

$$C_2 = \{00\ 00\ 00, 10\ 10\ 10, 01\ 01\ 01, 11\ 11\ 11\}.$$

This is an example of *repetition code.* Suppose that 110101 is received. Since this is not a codeword we can detect that at least one error has occurred. The codeword 010101 can be formed by changing one digit. Therefore we expect that 010101 was the most likely codeword transmitted, so we correct 110101 to 010101.

Now modify C_1 by adding a third digit to each codeword so that each codeword has an even weight. The resulting code is

$$C_3 = \{00\ 0, 10\ 1, 01\ 1\ , 11\ 1\}.$$

The added digit is called a *parity-check* digit. Suppose 010 is received, then since this is not a codeword we can detect that an error has occurred. Each of the codewords $110, 001$ and 011 can be formed by changing one digit in the received word. So in this case we can detect error but we are not sure what codeword to choose. We shall be able to overcome this problem but there will be times that a received word may not be corrected even with the most sophisticated decoding schemes. The next theorem shows us how many errors can be corrected in a code.

Theorem 1. *If d is the minimum distance of a code C, then C can correct $t = \lfloor (d-1)/2 \rfloor$ errors, and conversely.*

Proof. We prove that spheres of radius $t = \lfloor (d-1)/2 \rfloor$ about a codewords are disjoint. Suppose that they are not. Let x and y be distinct codewords in C, and assume that $v \in S_r(x) \cap S_r(y)$. Then by the triangle inequality, we have

$$d(x,y) \leq d(x,v) + d(v,y) \leq 2t \leq d-1.$$

Since $u + v \neq \theta$, we have $d(x,y) = wt(x+y) \geq d$. This contradiction shows that the spheres of radius t about codewords are disjoint.

♠ **The Information Rate**. The information rate or just *rate* of a code is a number that is designed to measure the proportion of each codeword that is carrying the message. The information rate of a binary code C of length n is defined to be

$$\frac{1}{n} \log_2 |C|.$$

Since $|C| \leq 2^n$, it is clear that the information rate ranges between 0 and 1; it is 1 if $C = \mathbb{K}^n$ and 0 if $|C| = 1$. For example, the information rates of the codes C_1,C_2, and C_3 in the previous section are 1, $1/3$, and $2/3$ respectively. Each of these information rates seems sensibly related to their respective codes, since the first 2 digits of the 6 in each codeword in C_2 can be consider to carry the message, as can the first 2 digits of the 3 in each codeword in C_3.

♠ **The Effect of Error Correction and Detection**. To exemplify the dramatic effect that the addition of a parity-check digit to a code can have in recognizing when error occur, we consider the following codes.

Suppose that all 2^{11} words of length 11 are codewords; then no error is detected. Let the reliability of the channel be $p = 1 - 10^{-8}$ and suppose that digits are transmitted at the rate of 107 digits per second. Then the probability that a word is transmitted with a single error is approximately $11p^{10}(1-p) \approx 11/10^8$. So about

$$\frac{11}{10^8} \times \frac{10^7}{11} = 0.1 \text{ words per second}$$

are transmitted incorrectly without being detected. That is one wrong word every second, 6 a minute, 360 an hour, or 8640 a day! Not too good.

Now suppose a parity-check is added to each codeword, so the number of 1's in each of the 2048 codewords is even. Then any single error is always detected, so at least 2 errors must occur if a codeword is to be transmitted incorrectly without our knowledge. The probability of at least 2 errors occurring is

$$1 - p^{12} - 12p^{11}(1-p) \approx \binom{12}{2} p^{10}(1-p)^2 \approx \frac{66}{10^{16}}$$

Now approximately $\frac{66}{10^{16}} \times \frac{10^7}{12} = 5.5 \times 10^{-9}$ words per second are transmitted incorrectly without being detected. That is about one error every 200 days!

So if we are willing to reduce the information rate by lengthening the code for 11 to 12 we are very likely to know when errors occur. To decide where these errors have actually occurred, we may need to request the retransmission of the message. It may be that retransmission is impractical, such as with the Voyager mission and when using compact discs. Therefore, at the expense of further increase in word length, it may well be worth incorporating error-correction capabilities into the code.

♣ **Maximum Likelihood Decoding**. Suppose we have an overall view of the transmission process, knowing both the codeword v that is transmitted and the word w that is received. For any given v and w, let $\Phi_p(v,w)$ be the probability that if the codeword v is sent over a $\mathcal{BSC}$ with reliability p, then the word w is received. Since we are assuming that noise is distributed randomly, we can treat the transmission of each digit as an independent event. So if $d(v,w)=d$, then we have $n-d$ digits correctly transmitted and d incorrectly transmitted and thus

$$\Phi_p(v,w) = p^{n-d}(1-p)^d.$$

Example. Let C be a code of length 5 and let $v = 10101$ and $w = 01101$). If $p = 0.9$, then

$$d(v,w) = wt(10101 + 01101) = wt(11000) = 2 \text{ with } \Phi_{0.9}(01101, 01101) = (0.9)^3(0.1)^2 = 0.00729.$$

In practice we know w but we don't know v. However each codeword v determines an assignment of probabilities $\Phi_p(v,w)$ to words w. Each assignment is a mathematical model and we choose the model which agrees most with observation. In this case, which makes the word received most likely. That is, assume v is sent when w is received if

$$\Phi_p(v,w) = \max\{\Phi_p(x,w) : x \in C\}.$$

The following theorem provides a criterion for finding such a codeword v.

Theorem 2. *Suppose we have a $\mathcal{BSC}$ with $1/2 < p < 1$. Let $v_1,\ v_2 \in C \subseteq I\!\!K^n$ and w, a received word with $d(v_1,w) = d_1$ and $d(v_2,w) = d_2$. Then*

$$\Phi_p(v_1,w) \le \Phi_p(v_2,w) \quad \textit{if and only if} \quad d_1 \ge d_2$$

Proof. Since $\dfrac{p}{1-p} > 1$, we have

$$\begin{aligned}\Phi_p(v_1,w) \le \Phi_p(v_2,w) &\Leftrightarrow p^{n-d_1}(1-p)^{d_1} \le p^{n-d_2}(1-p)^{d_2}\\ &\Leftrightarrow (\frac{p}{1-p})^{d_2-d_1} \le 1\\ &\Leftrightarrow d_2 \le d_1\,.\end{aligned}$$

This formally establish the procedure for correcting words: " correct w to a codeword which disagree with w in as few position as possible, since such a codeword is the most likely to have been sent, given that w has been received." Note that since $wt(v,w) = d(v,w)$, the equivalence statement in Theorem 1 may be replaced by the following:

$$\Phi_p(v_1,w) \le \Phi_p(v_2,w) \quad \text{if and only if} \quad wt(v_1,w) \ge wt(v_2,w)$$

We are now ready to give more precise formulation of two basic problems of Coding Theory.

♠ **Encoding**. We have to determine a code to use for sending our messages. We must make some choices. First we select a positive integer k, the length of each binary word corresponding to a message. k must be chosen so that $|M| \le I\!\!K^k = 2^k$, where M is the set of all messages. Next we decide how many digits we need to add to each word of length k to ensure that as many errors can be corrected or detected as we require; this is the choice of the codewords and the length of the code, n. To transmit a particular message, the transmitter finds the word of length k assigned to that message, then transmits the codeword of length n corresponding to that message of length k.

♠ **Decoding**. A word $w \in I\!\!K^n$ is received. We now describe a procedure, called *maximum likelihood decoding,* or $\mathcal{MLD}$, for deciding which codeword v in C was sent. There are actually two kinds of $\mathcal{MLD}$.

1) *Complete Maximum Likelihood Decoding,* or $\mathcal{CMLD}$. If there is one and only one codeword v of C closer to w than any other codeword in C, we decode w as v. That is $d(v, w) < d(x, w)$ for all $x \in C$, $x \neq v$, then decode w as v. If there are several codewords in C closest to v, i.e., the same distance from w, then we select arbitrary one of them and conclude that it was the codeword sent.

2) *Incomplete Maximum Likelihood Decoding,* or $\mathcal{IMLD}$. Again, if there is one and only one codeword v in C closer to w than any other codeword of C, we decode w as v. But if there are several codewords in C at the same distance from w, then we request a retransmission. In some cases we might ask for a retransmission if the received word w is too far away from any codeword of C.

The codeword v in C closest to the received word w has the greatest probability $\Phi_p(v, w)$ of being the word sent. Thus the strategy in $\mathcal{IMLD}$ is to examine the error pattern $u = v + w$ for all codeword v, and pick the codeword which yield the error pattern of smallest weight.