Channel Coding

Introduction

It is typically the case in digital communication systems that noise in the channel can cause errors in the received bit pattern. If the resulting *bit error rate*, or BER is unacceptable then we need a strategy to improve things. This leads to the idea of using *error correcting codes*.

Information and Entropy

Consider a random process or "source" called X that produces N possible signals. You can enumerate these symbols using $\log_2 N$ bits. For example, if your source produces two symbols (e.g., BPSK) you need 1 bit, i.e., your two symbols is labeled 0 or 1. If your source produces four symbols (e.g., QPSK) you need 2 bits, i.e., your four symbols are labeled 00, 01, 10, or 11. If each symbol is equally likely then we say that there are $\log_2 N$ bits of information per symbol. If the symbols are not equally likely then there are *less* than $\log_2 N$ bits per symbol. As a trivial example, if you have four possible symbols, but two of them occur with probability zero, i.e., never, then you really only need one bit to represent the two symbols that actually occur. In general, if p_k is the probability of occurrence for the k^{th} symbol, then the average number of bits of information per symbol is given by $H(X)$, the *entropy* of X:

$$\begin{aligned} H(X) &= \sum_{k=1}^{N} p_k \log_2 \frac{1}{p_k} \\ &= -\sum_{k=1}^{N} p_k \log_2 p_k \end{aligned} \tag{28.1}$$

It is easy to verify that $H(X) = \log_2 N$ for the equal probability case.

An important special case is the *binary source* in which two symbols are possible. If one symbol occurs with probability p, then the other must occur with probability $1-p$. The *binary entropy function* is then

$$H_2(p) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p} \tag{28.2}$$

This is plotted in Fig. 28.1.

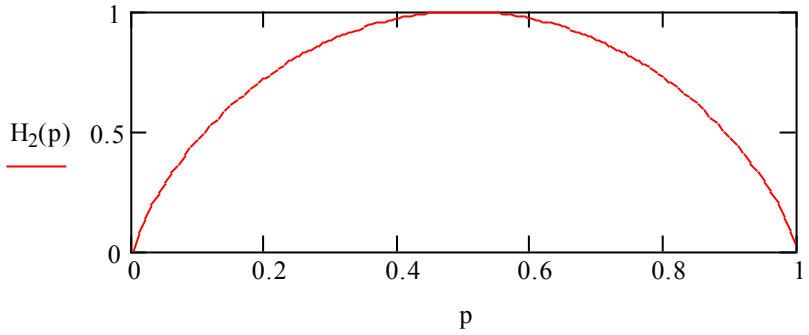

Figure 28.1: The binary entropy function. This gives the number of bits per symbol required, on average, to represent the output of a two-symbol source in which one symbol occurs with probability p and the other with probability $1-p$. *The entropy takes on its maximum value of 1 bit per symbol when* $p=0.5$

Channel Capacity

We've seen that for a binary channel we can ideally communicate one bit of information for every symbol transmitted, provided the two symbols (0 and 1) are equally free to occur. So we get one bit of information for one bit of transmission. Nothing surprising there. But now, suppose we are transmitting over a noisy channel. Noise, as we've seen, can cause bit errors. Let's say the bit error rate is P_e. For example, as we saw for BPSK/QPSK, $P_e = Q(\sqrt{2E_b/N_0})$. Having bit errors presumably reduces the information carrying "capacity" of the channel, but by how much? For each bit received there is a question: Is this bit in error or is it correct? The first possibility occurs with probability P_e and the second with probability $1-P_e$. Therefore the amount of information lost by not having an answer to this question is just the binary entropy function $H_2(P_e)$. It follows that the *channel capacity,* in bits of information per bit transmitted, is

$$\begin{aligned} C &= 1 - H_2(P_e) \\ &= 1 - P_e \log_2 \frac{1}{P_e} - (1-P_e)\log_2 \frac{1}{1-P_e} \\ &\underset{P_e \to 0}{\approx} 1 - P_e \log_2 \frac{1}{P_e} \end{aligned} \tag{28.3}$$

This is plotted in Fig. 28.2. When $P_e = 0.5$ the capacity falls to zero. This is the worst possible case and you cannot send any information over such a channel; each bit is completely randomized by noise. For lower bit error rates, however, the capacity is non-zero and it quickly approaches 1 as P_e decreases.

What does it mean to say that the information carrying capacity of a channel is something less than one bit per bit? We will find out when we discuss *error correcting codes* below.

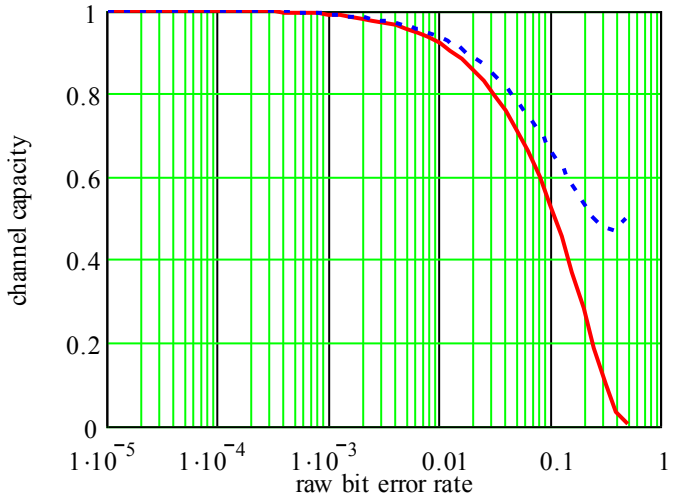

Figure 28.2: Channel capacity as a function of raw bit error rate. The solid curve is the exact result of (28.3). The dashed curve is the approximation.

First, let's take another look at the information carrying capacity of a channel. The following derivation is not rigorous, but you can think of it as a plausibility argument. Let W be the RF bandwidth of our channel and suppose our signal lasts for a time T. The Sampling Theorem tells us that this can be represented by $n = 2WT$ samples. Suppose the average signal power is P_S and the average noise power is $P_N = N_0 W$ where $N_0 = kT_N$ is the noise spectral density. Our n samples define a vector in an n-dimensional space. "Typical" signal voltages would be $V_S \propto \sqrt{P_S}$ while "typical" noise voltages would be $V_N \propto \sqrt{P_N}$. The noise would define an n-dimensional "noise sphere" of uncertainty about the signal point.

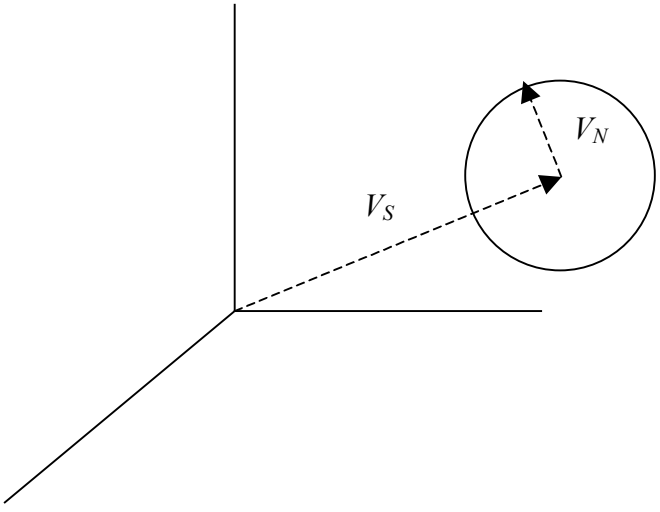

Figure 28.3: Signal as a vector in an n-dimensional space. Noise creates a sphere of uncertainty about the signal vector.

All possible signal plus noise vectors would fall within an *n*-dimensional hyper-sphere with radius of about $r = \sqrt{P_S + P_N}$. Since the "volume" of an *n*-dimensional hyper-sphere is proportional or r^n, the total volume available is proportional to $(P_S + P_N)^{n/2}$ while the volume occupied by a noise sphere is proportional to $P_N{}^{n/2}$. Therefore the number of noise spheres that can fit in the available volume is

$$M \approx \left(1 + \frac{P_S}{P_N}\right)^{n/2} \tag{28.4}$$

This is the maximum number of distinct signals, transmitted over a bandwidth W during a time T, that can be distinguished at the receiver. The number of bits of information is therefore

$$\begin{aligned} \log_2 M &= \frac{n}{2}\log_2\left(1 + \frac{P_S}{P_N}\right) \\ &= TW\log_2\left(1 + \frac{P_S}{P_N}\right) \end{aligned} \tag{28.5}$$

and the number of bits per second is

$$C = W\log_2(1 + S/N) \tag{28.6}$$

This form of the capacity is the maximum bit rate R_b at which the channel can carry information. Using the definition of spectral efficiency $\eta = R_b/W$, and using $P_S = E_b/T_b$, $R_b = 1/T_b$, and $P_N = N_0 W$, we can rearrange (28.6) to give

$$\left(\frac{E_b}{N_0}\right)_{\min} = \frac{2^\eta - 1}{\eta} \tag{28.7}$$

This gives the minimum E_b/N_0 that can support communication with spectral efficiency of η bits per second per Hz. This is plotted in Fig. 28.4. Note that η goes to zero for $E_b/N_0 = -1.6\,\text{dB}$. But note also that for $\eta = 1$, $E_b/N_0 = 0\,\text{dB}$.

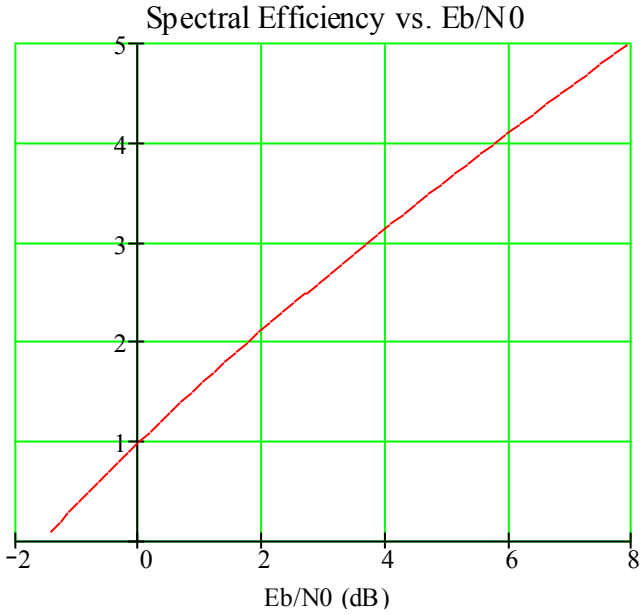

Figure 28.4: Theoretical maximum spectral efficiency for a given E_b / N_0.

Error Correcting Codes

Equation (28.3) tells us that we can never get a full bit of information for each transmitted bit. Instead, let's say we try to *code* every *block* of k bits of information into a block of n transmitted bits where $n \geq k$. We refer to this as an (n,k) code and we say that our code has a *rate* $R = k/n$.

As a simple example we could repeat every bit of data three times in our transmission. To communicate the data [0 1 0] we'd transmitted [000 111 000] and so on. In this case $k = 1$ and $n = 3$. We have a $(3,1)$ code and the rate is $R = 1/3$, that is, there is on average 1/3 of a bit of information in each transmitted bit. To see the usefulness of this, suppose we use no coding. If we have a bit error in our original data stream and [0 1 0] gets corrupted to [0 1 1], there is no way to realize this at the receiver. On the other hand, with coding, if [000 111 000] gets corrupted to [000 111 100], we can tell that the last three bits are not the same, as they should be. So, we know that the last "1" is an error and we can correct it to get the original transmitted signal [000 111 000]. From that we know that the data stream was [0 1 0]. This is a simple example of an *error correcting code.* The idea is to put some redundancy into the signal so that the receiver can identify and correct bit errors.

This is where (28.3) comes into play. In the late 1940's a fellow named Claude Shannon proved that given some P_e there must exist a one code with rate R arbitrarily close to the channel capacity C that allows error-free communication. Unfortunately this theorem doesn't find such a code it only proves existence. Since that time people have continued to come up with better and better codes that approach nearer to channel capacity C given in (28.3).

A useful geometric interpretation of block codes is as follows. For an (n,k) code, n bits of code are used to represent k bits of data. The n bits of code correspond to the 2^n vertices of an n-

dimensional binary "cube." For $n = 3$ we have a 3-dimensional cube as shown in Figure 28.5. The eight vertices represent all possible combinations of three bits: (0,0,0), (0,0,1), and so on. For the simple "repeat three times" code mentioned above, $k = 1$. So, we are trying to communicate one bit of data, and there are $2^1 = 2$ data states. We choose two of the eight vertices to represent these data states. In the example above we took the vertex (0,0,0) to represent data 0 and (1,1,1) to represent data 1. Now we can see where the error correcting capability comes from. A bit error moves us from a code vertex to an adjacent vertex. The code word (0,0,0) can change to (0,0,1), (0,1,0), or (1,0,0) while the code word (1,1,1) can change to (1,1,0), (1,0,1), or (0,1,1). As long as these two sets of adjacent vertices are distinct, we can correct a bit error.

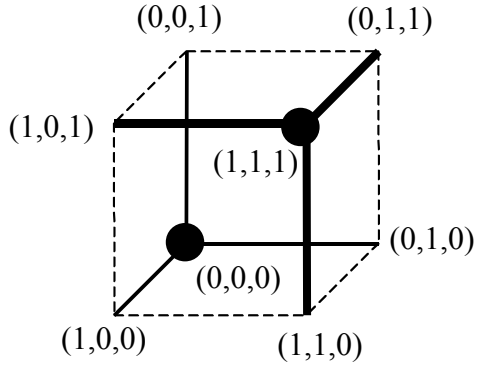

Fig 28.5: Geometric interpretation of a (3,1) error correcting code. The set of vertices with Hamming distance 1 from code word (0,0,0) is distinct from the set of vertices with Hamming distance 1 from code word (1,1,1).

The number of bits by which two sequences differ is called the *Hamming distance* between the sequences. For example, the Hamming distance between (0,0,1) and (0,1,1) is 1. The idea of block coding is this. Of the 2^n available vertices in an *n*-dimensional cube, choose 2^k to represent the possible sequences of *k* data bits. If the minimum distance between any two of these vertices is *d*, then this code can correct $m = (d-1)/2$ bit errors. In our simple (3,1) code the Hamming distance is 3, so we can correct 1 bit error.

Hamming Codes

Probably the simplest useful error correcting codes are the *Hamming codes*. These can be constructed as follows. Chose some integer *m*. Then $n = 2^m - 1$ and $k = n - m$. The code rate is

$$R = \frac{k}{n} = \frac{2^m - 1 - m}{2^m - 1} \tag{28.8}$$

Form an $m \times n$ *parity-check matrix H* in which the columns are all the possible non-zero, *m*-dimensional binary vectors. Arrange the columns so this has the form $H = [I_m Q]$ where I_m is the *m*-dimensional identity matrix. Then the *generator matrix* is $G = [Q^T I_k]$. The valid code words are the 2^k possible linear combinations of the rows of *G*.

As an example, let's take $m = 3$. Then $n = 7$ and $k = 4$, that is we have a (7,4) code and the rate is $R = 4/7$. The set of all possible non-zero, 3-dimensional binary vectors is

$$\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} \tag{28.9}$$

Then

$$H = \begin{bmatrix}1 & 0 & 0 & 1 & 1 & 0 & 1\\ 0 & 1 & 0 & 1 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 1\end{bmatrix} \tag{28.10}$$

and the generator matrix is

$$G = \begin{bmatrix}1 & 1 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 1 & 0 & 1 & 0 & 0\\ 0 & 1 & 1 & 0 & 0 & 1 & 0\\ 1 & 1 & 1 & 0 & 0 & 0 & 1\end{bmatrix} \tag{28.11}$$

Let **x** be a linear combination of the row vectors of G (all arithmetic is done modulo 2) such that the last k bits are identical to our data bits. Then the *syndrome* of **x**, $\mathbf{s} = H\mathbf{x}^T$ is zero. Let **y** be a corrupted version of **x** with a single bit error in the i^{th} element. Then $\mathbf{s} = H\mathbf{y}^T$ will be the i^{th} column of H. Therefore, if a single bit error occurs we can detect and correct it.

For example, suppose we want to transmit the data vector $\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix}$. Adding the first and fourth rows of G (modulo 2) gives $\mathbf{x} = \begin{bmatrix}0 & 0 & 1 & 1 & 0 & 0 & 1\end{bmatrix}$. The last four bits are our data and the first three bits can be considered *parity* bits. You can verify that the syndrome of **x** is **0**. Suppose in transmission this gets corrupted to $\mathbf{y} = \begin{bmatrix}0 & 0 & 1 & 1 & 0 & 1 & 1\end{bmatrix}$. Calculating the syndrome we find

$$\mathbf{s} = \begin{bmatrix}0\\1\\1\end{bmatrix} \tag{28.12}$$

This is the sixth column of H. Therefore we know that the sixth bit of **y** is in error (assuming only a single bit error has been made) and we can correct it to get **x**. We then extract the last four bits of **x** as our data.

Let p be the probability that any given bit is in error (what we've called P_e above). The probability that a given bit is correct is $(1-p)$. The probability that n bits are all correct is $(1-p)^n$. It follows that the probability that there are one or more bit errors in a sequence of n bits is $1-(1-p)^n$. The probability that there is one, and only one, bit error is $np(1-p)^{n-1}$. So the probability of two or more bit errors is the difference of these, or

$$P_{err} = 1-(1-p)^n - np(1-p)^{n-1} = \frac{n^2-n}{2}p^2 - \frac{n^3-3n^2+2n}{3}p^3 + \cdots \tag{28.13}$$

This is the probability our *k* data bits will not be perfectly communicated. In our example $n = 7$ so this probability is about $21p^2$. If we had sent the *k* bits of data uncoded, the probability of them not being perfectly communicated would be

$$P_{err} = 1-(1-p)^k = kp - \frac{k(k-1)}{2}p^2 + \cdots \tag{28.14}$$

For $k = 4$ this would be about $4p$. If p is very small then $21p^2 << 4p$ and we achieve much more reliable communication using the Hamming code.

Note from (28.3) that for a code rate 4/7 error-free communication is theoretically possible for a bit error rate of about $p = 0.088$. The (4,7) Hamming code does not achieve error-free communication for $p = 0.088$. Rather it lowers the probability of error for *k* bits from about 0.35 to about 0.16. This demonstrates how much more is theoretically possible. For this reason more complex block codes have been developed, most notably BCH and Reed-Solomon codes, that provide better performance.

References

1. McEliece, R. J., *The Theory of Information and Coding*, Cambridge University Press, 1977. ISBN 0-521-30223-4.
2. Lin, S., and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983. ISBN 0-13-283796-X.
3. Burr, A., *Modulation and Coding for Wireless Communications*, Prentice-Hall, 2001, ISBN 0-201-39857-5.
4. Pierce, J. R., *An Introduction to Information Theory,* Dover, 1980, ISBN 0-486-24061-4.