

8. Coding for Error Correction: Shannon's Second Theorem

8.1 Introduction

Now that we have examined such concepts as entropy of information and coding for compression, we are ready to consider the problem of correcting data that may have been corrupted. This is relevant when there is noise in data transmission or errors in our storage medium.

From here out, we will consider an M -bit message in which each bit is corrupted a proportion p of the time and the corruption of each bit is independent of that of other bits. This last part is false in practice as errors generally occur in groups in a message. For now we will ignore that.

Based on the above assumptions, we have a string of independent and identically distributed Boolean random variables corresponding to whether or not a bit is erroneous. This allows us to use the weak law of large numbers, as well as the binomial distribution.

Our goal now is to make our M -bit message longer, introducing some redundancy that will allow us to recognize when errors occur so we can recover our original message.

Two questions arise:

- 1) How much longer need our message be?
- 2) How can we construct a useful code for error correction?

Shannon's Second Theorem answers the first question.

Shannon's Second Theorem: Transmission of M message bits given a probability p of each bit being in error requires a proportion $H(p)$ of additional redundant bits, given large M . Thus, if we let the total number of bits needed be N , we have:

$$N = M + NH(p)$$

Almost all codes with bit length cM longer than this for any $c > 0$ allow recovery of the message, for a proportion of time that approaches 1.

Recall that $H(p) = -p \lg p - (1-p) \lg (1-p)$.

We will answer the second question after proving the first.

The theorem implies that choosing code words at random for each potential message would most likely allow us to recover the message, provided that the code words are of adequate length. However, there is no general method of finding such codes for large M and large ranges for p .

This seems like it should not be a practical problem, since we could simply generate a random set of code words and use them whenever message transmission is necessary.

There are three problems with that though.

First, when M is large, the number of potential messages 2^M is far too large to choose among them independently for each possible message. We could break the messages in to blocks of size k , but this becomes equally hopeless when the block sizes are about 50 or more.

The second problem is that encoding and decoding would require table lookup, which is a costly procedure in terms of storage of the code words and time to access the table.

This is related to the third problem, which is that, the message to decode will not even be in the table, since we assume some corruption. By the law of large numbers, approximately pN bits will be corrupted. While techniques exist to find the appropriate code word in the table, they are also costly and cumbersome.

Thus our goal is to find a code that allows simple encoding and decoding, even in the presence of errors. We will examine this in the next section of notes. Note that accomplishing this for large blocks makes this increasingly difficult. Thus, for this problem, random constructions are actually better than those that we devise ourselves, although we can't feasibly use them.

As an interesting side note, this conundrum is analogous to an interesting historical phenomenon. Through the Nineteenth and Twentieth Century, many learned individuals postulated that a planned economy would be far more efficient than a chaotic unstructured one, built up by the independent actions of individuals.

Putting structured economies into practice turned out to be disastrous in many cases, however. Perhaps these economic disasters occurred because the construction of an economy is a similar problem to that of making an error-correcting code, in which random construction is better than planning.

8.2 Proof of the Shannon Bound

We want to prove that the proportion of redundant bits, called **check bits**, needed to decode our message successfully is at least $H(p)$. We assume that the only errors possible are those that involve a bit changing, either from 0 to 1 or 1 to 0. Addition or removal of bits is assumed impossible.

When decoding our message, realize that the message we assume to be the true message is the one that differs from the received message in the fewest places. The number of places in which they differ is called the **Hamming distance**. For example, the messages 101 and 100 have a Hamming distance of 1. The Hamming distance between the received word and true word is also the number of errors that have occurred. By the

law of large numbers, we can assume that this number is on the order of Np . Let's call it $Np + q$, where $q \ll Np$.

There are $C(N, Np + q)$ ways of making $Np + q$ errors in a message of length N , using simple combinatorics. In much the same way that we developed the first Shannon's Theorem in section five of the notes using entropy of information and Stirling's Formula, we can rewrite this quantity as $2^{((N * H(p + (q/N)))(1 + o(1)))}$. The algebraic manipulation is not challenging and is left as an exercise.

Based on our assumptions, each possible pattern of errors is equally likely. Since there are 2^M possible messages sent and each can be corrupted using any of the error patterns, the number of possible received messages is $2^{M + ((N * H(p + (q/N)))(1 + o(1)))}$.

Note that there are not 2^N possible sent messages even though the messages are N bits in length. This is because the N -bit code words are created for each of the M bit messages we want to send. If there were 2^N N -bit messages, then we would have defeated the entire purpose of encoding for error correction, since there would always be at least one possible code word with Hamming distance of zero to the received word. That code word would be assumed to be the sent word, which is the same as the received word. In other words, this would not allow any error correction.

Using the pigeonhole principle, the number of possible received N -bit messages must be as great as the number of received messages we just determined. That means N must be on the order of \log base 2 of the number of possible received messages, which equals $M + ((N * H(p + (q/N)))(1 + o(1))) \approx M + NH(p)$. Thus, we have verified the Shannon Bound.

As another way to look at this, realize that each of the 2^M messages will generate $2^{NH(p)}$ possible received messages.

8.3 Proof that random codes come close to the Shannon Bound

All true messages have length M , while the code words have length N . Since there are many more possible N -bit words than M -bit words, each N -bit code word is chosen randomly and independently. Each possibility has equal probability of being chosen. Also, error generation occurs independently with probability p on each bit.

To clarify, we will call the received error-containing N -bit messages R . The true decoding of this is the M -bit message Z , with N -bit code word $C(Z)$. $C(Z)$ will generally have a Hamming distance of Np from R , by the weak law of large numbers. Also, the code words $C(Z)$ are selected randomly, so a particular N -bit string has a 2^{-N} probability of being selected as particular message Z .

After R is received, our plan is to decode to the code word D that is closest to it in Hamming distance. If D is the same as $C(Z)$, then we have been successful, otherwise we have not.

We will go wrong when there is another code word closer to R than $C(Z)$ is. That is, the other code word must have Hamming distance less than $N(p+e)$, where e is a constant much less than p such that $N(p+e)$ is the actual Hamming distance from R to $C(Z)$. $V(N(p+e))$ shall represent the number of N -bit words within Hamming distance $N(p+e)$ of R . Thus, the probability that a given N -bit string has Hamming distance less than $N(p+e)$ is $V(N(p+e))/2^N$. The probability that an N -bit string will *not* be confused with R is $1 - V(N(p+e))/2^N$.

There are $2^M - 1$ other valid N -bit strings that we do not want R to be confused with, so we want the probability that none are within $N(p+e)$ Hamming distance. Since they are independent, we simply multiply the probabilities that each will not be confused with R . This probability is: $\left[1 - \frac{V(N(p+e))}{2^N}\right]^{2^M - 1}$.

$\left[1 - \frac{V(N(p+e))}{2^N}\right]$ can be approximated as $\exp\left(-\frac{V(N(p+e))}{2^N}\right)$, and thus $\left[1 - \frac{V(N(p+e))}{2^N}\right]^{2^M - 1}$ can be approximated as $\exp\left(-\frac{V(N(p+e))}{2^{N-M+1}}\right)$. This is our probability of success. We can simplify this further since $\lg[V(N(p+e))]$ is on the order of $NH(p+e)$ (prove as exercise). With this simplification, the probability of success is at least $\exp(-2^{-(N-M-NH(p+e))})$.

Since Shannon's Bound requires that $N \gg M + NH(p)$, then the power of e is close to 0, and the probability of success is close to 1. Thus, a random construction of a code approaches a success rate of 1, given enough bits.

Exercises

Exercise 1 Verify that $C(N, Np + q)$ is on the order of $2^{((N \cdot H(p+q/N))(1+o(1)))}$.

Exercise 2 Show that $\lg[V(N(p+e))]$ is on the order of $NH(p+e)$.

- Steven Kannan