Error Correcting Codes

Jacob Cameron, Abi Hall, and Marcus Chijoff

Zero Dimensional Symmetry Summer Project

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Overview

An *error correcting code* is any method of encoding a message such that you are able to both detect and fix errors that occur in transmission. They are useful in situations where it is not practical to retransmit data if an error is detected. For example, they were used by the Voyager spacecraft due to the amount of time it takes to send data back to Earth from other planets.[1]

A message is made up of different 'words', which can be thought of as vectors in $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field of q elements. If they are in $\mathbb{F}_2^n$, then they are known as binary code words. The code words can be encoded with redundant information using a code defined by its $[n, k, d]$ structure where n is the length of the encoded words, k is the length of the words to be encoded, and d is the minimum distance between any two code words. The distance between two code words is defined as the number of components in the vectors that have different values. You are able to correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors in each code word.

Linear codes are codes where any linear combination of code words is also a code word. They allow you to use linear algebra in order to have more efficient encoding and decoding. As any linear combination of code words is also a code word, the code may be represented by the span of a set of k vectors. You can use this to create a matrix known as a generating matrix which gives you a code word whenever it is multiplied by a word. This matrix can be put in the form $[I_k|A]$ which allows for the code words to be decoded easily. The matrix representing the transformation whose kernel is the code is known as the parity check matrix. This is used to correct errors in transmission. If the generator matrix is in the form above, then the parity matrix is simply $\left[-A^T|I_{n-k}\right]$.

Say that you have a linear code with parity matrix P and you were trying to send the code word x but received $z = x + e$, where e is some error. Performing the matrix multiplication gives you $Pz^T = P(x+e)^T = Px^T + Pe^T$. However, as x is in the kernel of P we have that $Px^T = \mathbf{0}$ and thus that $Pz^T = Pe^T \coloneqq s$. Using a pre-computed lookup table, we are able to find that s corresponds to the error e and thus we are able to find x as $x = z - e$. This method is known as *syndrome decoding*.

Hamming Codes

Hamming codes are a family of linear codes of the form $[2^r - 1, 2^r - r - 1, 3]$ for $r \geq 2$. We will be focusing in the $\mathbb{F}_2 - [7, 4, 3]$ code which has generating matrix:

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

And parity matrix:

$$P = \begin{bmatrix} 1&1&0&1&1&0&0 \\ 1&0&1&1&0&1&0 \\ 0&1&1&1&0&0&1 \end{bmatrix}$$

Its syndrome lookup table is:

Syndrome	Error
(000)	(0000000)
(001)	(0000001)
(010)	(0000010)
(100)	(0000100)
(101)	(0100000)
(110)	(1000000)
(111)	(0001000)

This code can be constructed by using the Fano Plane, shown in Figure 1.

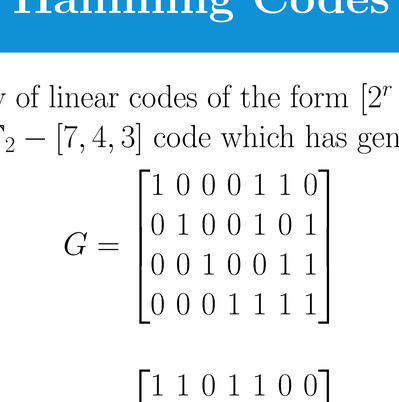

Fig. 1: The Fano Plane

The adjacency matrix of the Fano Plane — when reduced to row echelon form — is the generator matrix for the $\mathbb{F}_2 - [7, 4, 3]$ code. As this code has a distance of 3 it is able to correct up to $\left\lfloor \frac{3}{2} \right\rfloor = 1$ error in each code word. While this isn't a lot, the code adds relatively few bits to each message. This makes it good for situations where there isn't that much noise on the communication channel.

Example Encoding

Let's say you wanted to encode $m = (0110)$ with the $[7, 4, 3]$ code. You multiply it by G to get the code word.

$$mG = (0110110) \coloneqq c$$

Say that when this is transmitted, you receive $z = (1110110)$. Multiplying this by the parity matrix gives you:

$$Pz^T = (110)$$

Using the syndrome lookup table, we can see that this means an error occurred in the first bit. Thus, when we correct the error we get $z - (1000000) = (0110110) = c$.

As the generator matrix is in the form $[I_4|P]$, the first 4 components of the vector correspond to the original message. Thus, we can just take those first four components in order to decode the message.

Huffman Encoding

The ASCII standard encodes each of its supported 128 characters using eight bits. This is wasteful, as a message will most likely not use all 128 of the characters and some characters appear more often than others. For example, in the message "HelloWorld" there are three occurrences of the letter 'l' but only one of the letter 'd.' If we were to use a smaller number of bits to represent 'l' then we would have a much shorter message.

Huffman Encoding is a way of encoding a message such that the most common letters have shorter representations than the least common letters. It works by building a tree where each letter is assigned only to a leaf and the more common letters have a smaller distance to the root of a tree than the less common ones. Figure 2 shows a possible Huffman tree for the message "HelloWorld". Note that there can be more than one Huffman tree for a message.

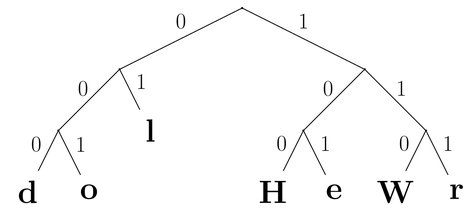

Fig. 2: A Possible Huffman Tree For The Message "HelloWorld"

In order to encode a message using a Huffman code, you traverse the tree from the root to the letter you want to encode. Going left corresponds to a "0;" going right corresponds to a "1." For example, using the tree in Figure 2, you would encode the letter "H" as "100." The encoding of the message "HelloWorld" is "10010101010011100011110100 0." In ASCII, "HelloWorld" is represented as "01001000 01100101 01101100 01101100 01101111 01010111 01101111 01110010 01101100 01100100." The Huffman code reduces the message to 33.75% of its size.

A tree is defined as any graph where there is exactly one path between any two vertices. As there is a unique path between any two vertices in a tree, there is one way to get from the root of a Huffman tree to each letter. Thus, you simply move along the encoded message until you are able to decode a letter by traversing the tree from the root to a leaf. You keep performing this process until you have reached the end of the message.

While Huffman encoding is not an error correcting code, they help other error correcting codes by reducing the length of the message, and thus the chance for it to be corrupted in transmission. However, if a Huffman encoded message is corrupted during transit, it will be much worse than a regular ASCII message. This is because changing a bit could result in the message decoding a letter with a different distance from the root. This will result the next block of the message having a different length, which can result in the error propagating much further in the message.

References

[1] Richard P. Laeser, William I. McLaughlin and Donna M. Wolff. 'Engineering Voyager 2's Encounter with Uranus'. In: *Scientific American* 255.5 (1986), pp. 36–45. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24976083.