Linear Independence and Coding : Hamming Code

Shweta Arya

Introduction:

Whenever there is a data transmission over a network from one source to another, there is a possibility of errors that might occur. Generally any image or message is sent out from one end of channel to another in form of bits. In this process the bits might get flipped and the received message will not be same as the sent one. Here comes the error detecting and correcting codes in picture.

Before sending the message we encode the message with additional bits applying some linear alzebra properties of subspaces. Such that when the message is received at the receivers end we can detect any error in the message based on extra bits added and further correct it using the linear alzebra properties. In this part we will cover how can we encode a message, detect and correct errors and decode it to get the correct message at receivers end. There are many ways of doing so

- Repetition Codes
- Hamming Codes
- Polynomial Codes

1. Repetition Codes- The idea of the repetition code is to just repeat the message several times. The assumption is that the channel might corrupt only minority of these repetitions. Thus the receiver will notice that a transmission error occurred since the received data stream is not the repetition of a single message, and moreover, the receiver can recover the original message by looking at the received message in the data stream that occurs most often. Example- Consider a binary repetition code of length 3. The user wants to transmit the information bits 101. Then on encoding we get the 111 000 111, which will be transmitted. Let's say three errors corrupt the transmitted bits and the received sequence is 111 010 101. Decoding is usually done by a simple majority decision for each code word. That lead us to 101 as the decoded information bits. But if received 111 010 100 then decoded message is 100. So repletion codes fails here.

2. Hamming Codes-Now lets discuss in detail about Hamming Code introduced by **Richard W. Hamming.**
Code: A code C of length n over an alphabet K is a subset of K^n = KxKx xK , the Cartesian product of K with itself n times, which consists of n-tuples from the elements of K. Ex- For B ={0,1} for length n we have B^n ,we can have code C={101,100} which is a subset of B^2.

Hamming distance: The Hamming distance d(a,b) between two vectors a and b is the number of bits they differ by. The Hamming weight wt(a) of a vector is the number of non-zero components it has.

Minimum distance: For a code C is min{ d(u,v) : u, v in C, u ≠ v}, that is the smallest of the distances between distinct pairs of codewords. Similarly, the minimum weight 8 of a code C is min{ wt(u) : u in C, u ≠ the zero vector}, the smallest weight among non-zero codewords. The minimum distance of a code determines its error detection/correction capacity. So if we have a code C with minimum distance d. Then C can detect any d-1 errors, and can correct any (d-1)/2 (floor value) errors.

The first mathematical observation in the direction of linear alzebra for getting a mathematical structure is to notice **that B^n is a vector space over the binary field B**. Since the binary linear codes are subset of B^n, so binary linear codes of length n (linear codes in general for K^n) form a vector subspace of B^n. A code is called linear if it can be identified with a linear subspace of some finite-dimensional vector space(as identified above). A linear code followes following properties- The zero vector is a codeword, the sum of any two codewords is a codeword ($c(x) + c(y) = c(x + y)$) and any scalar multiple of a codeword is a codeword. In our report, all of our vector spaces will be $\{ 0,1 \}^n$, that is tuples of bits under addition mod 2.

A linear code C that has length n, dimension k, and minimum distance d is referred to as an [n,k,d]-code. **In mathematical terms, Hamming codes are a class of binary linear codes**. For a linear code C, its minimum distance is equal to its minimum weight. Minimum distance of the Hamming code is 3. Therefore, it is a [7,4,3]- code. It can detect any (d-1=3-1=2)double errors and correct any ((d-1)/2)single errors.

There are other ways of obtaining or defining vector spaces from linear algebra. For example, the row space of a matrix can be used to define a vector space. Similarly a null space can also be used. So we can define linear codes with help of matrices.

Generator Matrix and Parity Matrix- A generator matrix of a linear code C is a matrix G whose rows span C. Therefore, the code (of dimension k) can be defined as either $C = \{u * G : u \; in \; B^k\}$. A parity check matrix H of a linear code is a matrix whose null space is C, so the code (of dimension k) can be also defined as $C = \{u \; in \; B^n : H * u = 0 \; vector\}$. The rank of G or the nullity of H give the dimension of C.

[7,4] Hamming code

In 1950, Hamming introduced the [7,4] Hamming code. It encodes four data bits into seven bits by adding three parity bits. It can detect and correct single-bit errors. Lets see how generator matrix and parity Matrix are formulated and error detection and correction is done.

Encoding: Traditional Hamming codes are (7, 4) codes, encoding four bits of data into seven bit blocks (a Hamming code word). The extra three bits are parity bits. Each of the three parity bits are calculated based on parity of the 3 of the 4 data bits. All of the parity bits are even parity. Example: Given: data bits d1, d2, d3, and d4 then A (7, 4) Hamming code may define parity bits p1, p2, and p3 as p1 = d2 + d3 + d4, p2 = d1 + d3 + d4 ,p3 = d1 + d2 + d4 also there's a fourth equation for a parity bit that might be used in Hamming codes: p4 = d1 + d2 + d3. A valid Hamming codes may use any three parity bit definitions. Valid Hamming codes may also place the parity bits in any location within the block of 7 data and parity bits. Two Hamming codes with different parity bits or parity bits in a different bit position are considered equivalent. They will produce different results, but they are still Hamming codes.

One method for transforming four bits of data into a seven bit Hamming code word is to use a 4×7 generator matrix [G].Lets define d to be the 1×4 vector [d1 d2 d3 d4]. It's possible to create a 4×7 generator matrix [G] such that we get the desired 1×7 Hamming code word. Here's how it's done:

Step 1. We represent each data bit with a column vector as follows:

$$d1= \begin{vmatrix}1\\0\\0\\0\end{vmatrix} \quad d2= \begin{vmatrix}0\\1\\0\\0\end{vmatrix} \quad d3= \begin{vmatrix}0\\0\\1\\0\end{vmatrix} \quad d4 = \begin{vmatrix}0\\0\\0\\1\end{vmatrix}$$

Step 2. We represent each parity bit with a column vector containing a 1 in the row corresponding to each data bit included in computation based on parity bit definitions above.

$$p1= \begin{vmatrix}0\\1\\1\\1\end{vmatrix} \quad p2= \begin{vmatrix}1\\0\\1\\1\end{vmatrix} \quad p3= \begin{vmatrix}1\\1\\0\\1\end{vmatrix}$$

Step 3. Create a generator matrix, [G], by arranging the column vectors from the previous steps into a 4×7 matrix such that the columns are ordered to match their corresponding bits in a code word. To create a generator that produces code words with the bits ordered p1, p2, p3, d1, d2, d3, d4 (3 parity bits followed by 4 data bits) use the vectors from the previous steps and arrange them into the following columns [p1 p2 p3 d1 d2 d3 d4], arranging the columns differently will just change the position of bits in our code word.

So the matrix G= $\begin{bmatrix}0&1&1&1&0&0&0\\1&0&1&0&1&0&0\\1&1&0&0&0&1&0\\1&1&1&0&0&0&1\end{bmatrix}$. So $C = \{u * G : u \; in \; B^k\}$, where u is the data vector.

Error detection and correction: When the encoded data is received we need to check the parity values to verify if the data received is correct or not. Parity is validated using matrix operations. A 3×7 parity check matrix [H] may be constructed such that row 1 contains 1s in the position of the first parity bit and all of the data bits that are included in its parity calculation. Row 2 contains 1s in the position of the second parity bit and all of the data bits that are included in its parity calculation, similarly for row 3. Using the code from example above, the matrix H may be defined as follows:

H=$\begin{bmatrix}1&0&0&0&1&1&1\\0&1&0&1&0&1&1\\0&0&1&1&1&0&1\end{bmatrix}$, $C = \{u \; in \; B^n : H * u = 0 \; vector \; or \; z \; a \; syndrome \; vector\}$, u is the received codeword.

Multiplying the 3×7 matrix [H] by a 7×1 matrix representing the encoded data produces a 3×1 matrix called the "syndrome". There are two useful proprieties of the syndrome. If the syndrome is all zeros, the encoded data is error free. If the syndrome has a non-zero value, flipping the encoded bit that is in the position of the column in [H] that matches the syndrome will result in a valid code word.

Example: Lets encode the data value 1010 using the Hamming code defined by the matrix G (above)

uG = C .

$$\begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = [1\ 0\ 1\ 1\ \ 0\ 1\ 0]$$ -encoded codeword

Using the parity check we can correct and verify the received code word 1011011 for the above sent data Hu = Syndrome vector.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$=[1 1 1] the syndrome vector we got is non-zero so there is a parity error.

Looking back at the matrix [H], w will see that the seventh column(binary equivalent of 111,so the seventh bit is the errored bit. Changing the seventh bit produces the code word 1011010.

3. Polynomial Code- A polynomial code is a type of linear code whose set of valid code words consists of polynomials that are divisible by a given fixed polynomial .