PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MAY 2020: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE18MA251- LINEAR ALGEBRA

MINI PROJECT REPORT
ON
LINEAR ERROR CORRECTING CODES

Submitted by

1.	ABDUR RAHMAN HATIM	PES1201801503
2.	HARICHANDANA MAGAPU	PES1201801041
3.	SANJNA CHATURVEDI	PES1201800722

Branch & Section : CSE A

PROJECT EVALUATION
(For Official Use Only)

S No.	Parameter	Max Marks	Marks Awarded
1	Background & Framing of the problem	4	
2	Approach and Solution	4	
3	References	4	
4	Clarity of the concepts & Creativity	4	
5	Choice of examples and understanding of the topic	4	
6	Presentation of the work	5	
	Total	25	

Name of the Course Instructor : Nagegowda KS

Signature of the Course Instructor :

CONTENTS

Introduction

What are error correcting codes?

Digital devices are constantly sending and receiving signals. From cell phones communicating with each other, all the way to a command station trying to communicate to its space station. These signals often have to travel through various channels to get from one user to the other. But while travelling, there is a very high possibility of the message being interfered with because of an external disturbance. This external disturbance is referred to as noise. Once a message has been distorted, there is no possible way for the receiver to verify if the message they received is accurate or not. This is why error correcting codes were developed.

Error correcting codes are used by both the sender and the receiver to encrypt the message on the sender's side, and then decrypt the message on the receiver's side. This sounds very similar to normal cryptography, but there is a key difference; cryptographic encryption is designed so as to prevent an external attacker from accessing the message, whereas an error correcting code is designed to allow the receiver to identify possible errors that may have occurred during transmission.

A naive error correcting code would merely involve sending each bit in the message twice.
For example if the original message is 1010.
The encrypted message (massage being transmitted) would be 11|00|11|00.
As the receiver is also aware of the mechanism being used, they will be able to identify if an error occurred in the message.
If the message received was 11|01|11|00.
The receiver will be able to identify where the error is located. Here in the second segment of the message. But now lies another problem. The receiver has no way of knowing what the actual bit is. Is the segment supposed to contain a 0 or a 1.
In order to both identify and correct errors, we must pass at least 3 copies of one bit.
The encrypted message would now be 111|000|111|000.

Now when an error occurs in the second segment 111|010|111|000.
It is far more likely that the only one bit was changed (the 2nd 0 was swapped to a 1). The receiver will now be able to extract the original message despite any noise interference that may have occurred.

There are however several drawbacks to this approach. The first being, one needs to send 3 times as much information. And the second being there is a compromise on the speed with which one can send messages. To try to mitigate some of these problems, we saw the creation of linear codes, such as Hamming and Golay Codes.

Applications

Error correcting codes have many applications in multiple vast and diverse fields such as:

Deep Space Telecommunications: Development of error-correction codes was tightly coupled with the history of deep-space missions due to the extreme dilution of signal power over interplanetary distances, and the limited power availability aboard space probes. Whereas early missions sent their data uncoded, starting from 1968, digital error correction was implemented in the form of (sub-optimally decoded) convolutional codes and Reed:Muller codes. The Voyager 1 and Voyager 2 missions, which started in 1977, were designed to deliver colour imaging amongst scientific information of Jupiter and Saturn. This resulted in increased coding requirements, and thus, the spacecraft were supported by (optimally Viterbi-decoded) convolutional codes that could be concatenated with an outer Golay (24,12,8) code. The CCSDS currently recommends usage of error correction codes with performance similar to the Voyager 2 RSV code as a minimum. Concatenated codes are increasingly falling out of favor with space missions, and are replaced by more powerful codes such as Turbo codes or LDPC codes.

Data Storage: Error detection and correction codes are often used to improve the reliability of data storage media. A "parity track" was present on the first magnetic tape data storage in 1951. The "Optimal Rectangular Code" used in group coded recording tapes not only detects but also corrects single-bit errors. Some file formats, particularly archive formats, include a checksum (most often CRC32) to detect corruption and truncation and can employ redundancy and/or parity files to recover portions of corrupted data. Reed Solomon codes are used in compact discs to correct errors caused by scratches.
Modern hard drives use CRC codes to detect and Reed:Solomon codes to correct minor errors in sector reads, and to recover data from sectors that have "gone bad" and store that data in the spare sectors.[12] RAID systems use a variety of error correction techniques to correct errors when a hard drive completely fails. Filesystems such as ZFS or Btrfs, as well as some RAID implementations, support data scrubbing and resilvering, which allows bad blocks to

be detected and (hopefully) recovered before they are used.[13] The recovered data may be re-written to exactly the same physical location, to spare blocks elsewhere on the same piece of hardware, or the data may be rewritten onto replacement hardware.

The Internet: TCP provides a checksum error correction for protecting the payload and addressing information from the TCP and IP headers. Packets with incorrect checksums are discarded within the network stack, and eventually get retransmitted using Automatic Repeat Request, either explicitly (such as through triple-ack) or implicitly due to a timeout.

Terminology

Let A = $\{a_1, a_2, ..., a_q\}$ be a set of size q, which we refer to as a code alphabet and whose elements are called code symbols.

a) A q-ary **word** of length n over A is a sequence $w = w_1w_2 \cdots w_n$ with each $w_i \in A$ for all i. Equivalently, w may also be regarded as the vector $(w_1, \cdots, wn)$.

b) A q-ary **block code** of length n over A is a non-empty set C of q-ary words having the same length n.

c) An element of C is called a **codeword** in C

d) The number of codewords in C, denoted by |C |, is called the **size** of C.

e) The (information) **rate** of a code C of length n is defined to be$(\log_q|C|)/n$.

f) A code of length n and size M is called an (n, M)-code.

Decoding Rule

In a communication channel with coding, only codewords are transmitted. Suppose that a word w is received. If w is a valid codeword, we may conclude that there is no error in the transmission. Otherwise, we know that some errors have occurred. In this case, we need a rule for finding the most likely codeword sent. Such a rule is known as a decoding rule.

Maximum likelihood decoding:

Suppose that codewords from a code C are being sent over a communication channel. If a word x is received, we can compute the forward channel probabilities

P (x received | c sent)

for all the codewords $c \in C$. The maximum likelihood decoding (MLD) rule will conclude that c_x is the most likely codeword transmitted if c_x maximizes the forward channel probabilities; i.e.,

P (x received | c_x sent) = max $_{c \in C}$ P (x received | c sent).

There are two kinds of MLD:

(i) Complete maximum likelihood decoding (CMLD): If a word x is received, find the most likely codeword transmitted. If there are more than one such codewords, select one of them arbitrarily.
(ii) Incomplete maximum likelihood decoding (IMLD): If a word x is received, find the most likely codeword transmitted. If there are more than one such codewords, request a retransmission.
For a code C containing at least two words, the (minimum) **distance** of C, denoted by d(C), is $d(C) = \min \{d(x, y): x, y \in C, x \neq y\}$.
A code of length n, size M and distance d is referred to as an (n, M, d)-code. The numbers n, M and d are called the parameters of the code.
The distance of a code is intimately related to the error-detecting and error-correcting capabilities of the code.
Let v be a positive integer. A code C is v-error-correcting if minimum distance decoding is able to correct v or fewer errors, assuming that the incomplete decoding rule is used. A code C is exactly v-error-correcting if it is v-error-correcting but not (v + 1)-error-correcting.
A code C is v-error-correcting if and only if $d(C) \geq 2v + 1$; i.e., a code with distance d is an exactly $\lfloor (d-1)/2 \rfloor$-error-correcting code. Here, $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

Vector Spaces over a Finite Field

Let F_q be the finite field of order q. A nonempty set V, together with some (vector) addition + and scalar multiplication by elements of F_q,is a vector space (or linear space)over F_q if it satisfies all of the following conditions. For all $u,v,w \in V$ and for all $\lambda,\mu \in F_q$:

1. $u+v \in V$;
2. $(u+v) + w = u+ (v + w)$;
3. There is an element $0 \in V$ with the property $0+v = v = v+0$ for all $v \in V$;
4. For each $u \in V$ there is an element of V, called $-u$, such that $u+(-u) = 0 = (-u) + u$;
5. $u+v=v+u$;
6. $\lambda v \in V$;
7. $\lambda(u+v) = \lambda u+\lambda v$, $(\lambda+\mu) u=\lambda u+\mu u$;
8. $(\lambda\mu)u = \lambda(\mu u)$;

9. If 1 is the multiplicative identity of F_q, then 1u=u.

A **linear code** C of length n over F_q is a subspace of $\mathbf{F}_q^n$

Hamming weight

Let x be a word in $\mathbf{F}_q^n$. The (Hamming) weight of x, denoted by wt(x), is defined to be the number of nonzero coordinates in x; i.e.,

$$wt(x) = d(x,0)$$

Bases for Linear Codes

Since a linear code is a vector space, all its elements can be described in terms of a basis. There are three algorithms that yield either a basis for a given linear code or its dual:

Algorithm 1

Input: A nonempty subset S of $\mathbf{F}_q^n$.

Output: A basis for C = < S >, the linear code generated by S.

Description: Form the matrix A whose rows are the words in S. Use elementary row operations to find an REF of A. Then the nonzero rows of the REF form a basis for C.

Algorithm 2

Input: A nonempty subset S of $\mathbf{F}_q^n$.

Output: A basis for C = < S >, the linear code generated by S.

Description: Form the matrix A whose columns are the words in S. Use elementary row operations to put A in REF and locate the leading columns in the REF. Then the original columns of A corresponding to these leading columns form a basis for C.

Algorithm 3

Input: A nonempty subset S of $\mathbf{F}_q^n$.

Output: A basis for the dual code $C^{\perp}$ where C = < S >.

Description: Form the matrix A whose rows are the words in S. Use elementary row operations to place A in RREF. Let G be the k × n matrix consisting of all the nonzero rows of the RREF:

$$A \rightarrow \begin{pmatrix} G \\ O \end{pmatrix}.$$

(Here, O denotes the zero matrix.)

The matrix G contains k leading columns. Permute the columns of G to form

$$G' = (I_k|X),$$

where I_k denotes the k × k identity matrix. Form a matrix H′ as follows:

$$H' = \left(-X^{\mathrm{T}}|I_{n-k}\right),$$

where X^T denotes the transpose of X.
Apply the inverse of the permutation applied to the columns of G
to the columns of H′ to form H. Then the rows of H form a basis for $C^{\perp}$

For a q-ary code C with parameters (n, M, d), the **relative minimum distance** of C is defined to be δ(C) = (d − 1)/n.

For any code C over F_q, the **extended code** of C, denoted by $\overline{C}$, is defined to be

$$\overline{C} = \left\{ \left(c_1, \ldots, c_n, -\sum_{i=1}^{n} c_i \right) : (c_1, \ldots, c_n) \in C \right\}.$$

When q = 2, the extra coordinate $-\sum_{i=1}^{n} c_i$ added to the codeword $(c_1, \ldots, c_n)$ is called the parity-check coordinate.

Hamming Bound

For an integer q > 1 and integers n, d such that $1 \leq d \leq n$, we have:

$$A_q(n,d) \leq \frac{q^n}{\sum_{i=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{i}(q-1)^i}$$

A q-ary code that attains the Hamming (or sphere-packing) bound, is called a **perfect code**. Some of the earliest known codes, such as the **Hamming codes** and the **Golay codes**, are examples of perfect codes.

Hamming Code

Hamming codes have the ability to correct 1 bit errors and to detect up to 2 bit errors. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance of three. [1.] Hamming codes can only detect and correct errors when the error rate is low. There are two approaches we have used to solve hamming codes. This first is a linear algebra approach and the second makes use of parity bits.

Linear Algebra Approach to solving Hamming Codes

Defining the vector space.

Let 0 represent all even numbers, and 1 represent all odd numbers.

Let $Z^n{}_2$ be a vector space of 2 containing {0 , 1}

$Z^n{}_2 = \{[x_1,x_2,\dots,x_n] : x_1,x_2,\dots.x_n \in Z \}$

$Z^n{}_2$ is not infinite like R^n. It is a finite vector space.

The number of vectors in $Z^n{}_2$ is 2^n. As each value in the unit vector can have one of two possible values. That is either a 0 or a 1.

The number of nonzero vectors in Z_n is $Z^n - 1$. This is because there is only one zero vector.

For example, $Z^3{}_2$ where n = 3, has 7 nonzero vectors.

Defining the Hamming Matrix

A Hamming Matrix is a matrix H with K rows and the columns of H all the nonzero vectors in $Z^k{}_2$. Here, k is a non-negative integer.

For example, k = 3, there are 7 nonzero vectors.

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

The order in which the columns are written does not matter. But we have chosen to write it this way as now, the first 3 columns are the identity matrix of k, here 3. And the rest of the matrix is some 3 x 4 matrix Q.

H = [I(3) Q]

In the general case, H = [I(k)   Q]. Here, Q is a K x (2^k-1-k) matrix.

Performing a linear transformation on H to obtain matrix M.

$$M = \begin{bmatrix} Q \\ I(4) \end{bmatrix}$$

In the general case, M has 2^k-1 rows and 2^k-1-k columns.

$$M = \begin{bmatrix} Q \\ I(l) \end{bmatrix}$$

l will be a K x (2^k -1 - k) matrix.

Characteristics of H and M

1. H is a surjective matrix

   Proof:

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

This is in reduced echelon form. There are 3 pivots, one for every row.

Therefore H is a surjective matrix.

In the general case.

$$H = [I(k) \quad Q]$$

Here, there will be 1 pivot for every row as I(k) is the identity matrix of order(k x k).

1. M is injective

Proof: For a matrix to be injective, its null space must be zero. That is, all of its columns must be linearly independent.

$$M = \begin{bmatrix} Q \\ I(4) \end{bmatrix}$$

Here, we see that the bottom matrix is an identity matrix, this implies that every column will be linearly independent.

In the general case, $M = \begin{bmatrix} Q \\ I(l) \end{bmatrix}$

As the lower matrix is the identity matrix, it implies that every column will be linearly independent.

1. Karnal (Null space) of H is the Image (column space) of M.

Proof:

For, Kar(H) = Image(M)

Kar(H) $\subseteq$ Image(M) and Kar(H) $\supseteq$ Image(M)

First, to prove that Kar(H) ⊆ Image(M).

$$\begin{bmatrix} Q \\ I(4) \end{bmatrix}$$

The product of H and M, HM = [I(3) Q]

= I(3).Q + Q.I(4) = 2[Q]

2 is an even number, so 2 => 0

HM = $0_{(3 \times 4)}$

This proves that the image is a subset of the karnal.

Secondly, to prove that Kar(H) Image(M)

As M is injective, the dimension of the image will be 4.

dim(Image(M)) = 4

As H is surjective, the dimension of the image of H will be 3.

dim(Image(H)) = 3

dim(Kar(H)) = 4

As 2 different subspaces have the same dimension and one is contained within the other, we can infer that the 2 subspaces are equal.

Properties:

Let Image(M) = Kar(H) = C

Let C_i be the subspace C shifted by the ith vector

That is $C_i = C + \mathbf{e}_i$

For H = I$(_{3)}$ $_{Q}$ and M = $\begin{bmatrix} Q \\ I(4) \end{bmatrix}$

$C_i = \mathbf{V}\ C$ where i ∈ {1,2,..,7}; the basis of vectors in Z^3_2.

Claim: C is the solution of a homogenous system.

1. C_i is also the solution set of an inhomogeneous system and the solution will be $H\mathbf{e_i}$ and will be the i^th^ column of H.
2. Talking any 2 subsets of C_i and C_j will yield disjoint vectors:

   $C_i \cap C_j = \Phi$ for all $i \neq j$
3. Each subset is disjoint from the homogeneous solution set. $C_i \cap C_j = \Phi$ This implies that the Z^n_2 is the sum of C and all C_i for every value of i. The entire vector space is the union of all these vectors.

Proving Claims:

C_i is also the solution set of an inhomogeneous system and the solution will be $H\mathbf{e_i}$ and will be the i^th^ column of H.

$\mathbf{e_i}$ is a particular solution.

Therefore the solution set of the whole system $H\mathbf{x} = H\mathbf{e_i}$ is $\mathbf{e_i} + C$

$C_i \cap C_j = \Phi$ for all $i \neq j$

Consider, $M\mathbf{u_1} + \mathbf{e_i} = M\mathbf{u_2} + \mathbf{e_j}$

Here, $M\mathbf{u_1}$ is C_i and $M\mathbf{u_2}$ is C_j

Applying H on both sides,

$H M\mathbf{u_1} + H\,\mathbf{e_i} = H M\mathbf{u_2} + H\,\mathbf{e_j}$

$H M\mathbf{u_1} = H M\mathbf{u_2} = 0$

$e_j + e_i \quad i = j$

The entire vector space is the union of all these vectors.

Prof:

To prove that $Z^4_2 = C \cup \bigcup_{i=1}^{7} C_i$

Number of vectors in C is 2^4

Number of vectors in C_i is the same as the number of vectors in C (C_i is essentially C but shifted). Which means there are 2^4 vectors in C_i.

The number of vectors in $C \cup \bigcup_{i=1}^{7} C = 2^4 + 7 * 2^4 = 2^7$

And the number of vectors in $Z^4{}_2$ is 2^7

Form proof 1 and 2, we can infer that $Z^4{}_2 = C \cup \bigcup_{i=1}^{7} C$

Significance:

H can distinguish between vectors in C versus vectors in C_i

Because HC = { **0** } and $HC_i = \{H\mathbf{e_i}\}$

Applying H to an arbitrary vector V $\in$ Z will result in it equalling 0 or one of the columns.

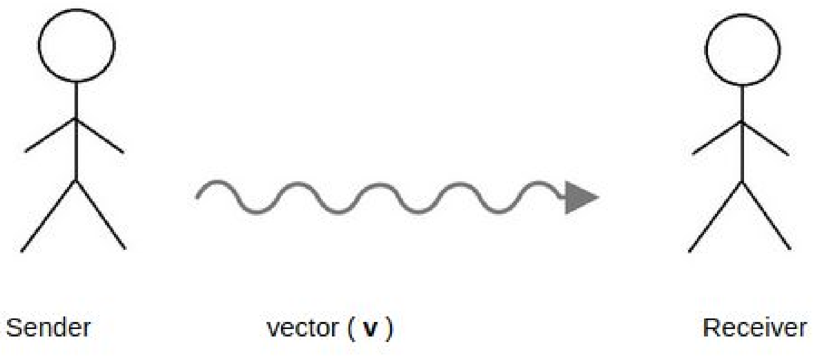
If H**V** = 0 then **V** $\in$ C

And if HV = $H\mathbf{e_i}$ then **V** $\in C_i$

This ability of H being able to distinguish between the C and the various C_i is why we can apply this mathematical model to error detecting codes.

Example:

A message is being sent from a sender to receiver through a vector **v**, there is a possibility that some of the elements in **v** have been altered. Both sender and receiver have agreed to use Hamming codes as a common model of encryption and decryption.



A copy of the original message its in **v**

Let M**u** = $\begin{bmatrix} Qu \\ u \end{bmatrix}$ for all **u** Z^4_2

Where M**u** C and M**u** Z^7

V = M**u** + $\mathbf{e}_i$

If

i) H**v** = **0** **v** C = Image(M)

ii) H**v** = H$\mathbf{e}_i$ **v** $C_i = C + \mathbf{e}_i$

Assume at most one error.

In case i)

original message is the last 4 entries **v**

In case ii)

Error occurred in the ith entry of **v** + $\mathbf{e}_i$

Then the error occurred in the ith entry of **v** to fix **v** + $\mathbf{e}_i$. This would make the last 4 entries of **v** + $\mathbf{e}_i$ the original message.

V = $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

H**v** = $\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

= $\begin{bmatrix} 3 \\ 2 \\ 3 \end{bmatrix}$ = $\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$ This is the 6th column.

Which means that the original message is $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

*What Q**u** is doing?*

Parity approach to solving Hamming Codes

M = $\begin{bmatrix} Q \\ I(4) \end{bmatrix}$ and **u** = $\begin{bmatrix} u1 \\ : \\ : \\ u2 \end{bmatrix}$

Q**u** = $\begin{bmatrix} u1 + u2 + u3 \\ u1 + u2 + u4 \\ u1 + u2 + u3 \end{bmatrix}$ = $\begin{bmatrix} P1 \\ P2 \\ P3 \end{bmatrix}$

The P1, P2, and P3 are the called parity bits.

$$\mathbf{V} = \begin{bmatrix} P1 \\ P2 \\ P3 \\ U1 \\ U2 \\ U3 \\ U4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Comparing each parity bit to see if it matches

$P_1 = 0 = U_1 + U_3 + U_4$

$= 1 + 1 + 1 = 3 = 1$

Which is inconsistent.

$P_2 = 0 = U_1 + U_2 + U_4$

$= 1 + 0 + 1 = 2$

$= 0$

Which is consistent.

$P_3 = 1 = U_1 + U_2 + U_3$

$= 1 + 0 + 1 = 2$

$= 0$

Which is inconsistent.

This means the U_3 is the incorrect bit.

The corrected message is $\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$

Thus proving that both the linear algebra approach and the parity bit approach yield the same result.

Golay Code

Extended Binary Golay Code:

A convenient way of finding a binary [23, 12, 7] Golay code is to construct first the extended Golay [24, 12, 8] code, which is just the [23, 12, 7] Golay code augmented with a final parity check in the last position.

Let G be the 12 × 24 matrix G = [I_{12} | A], where I12 is the 12 × 12 identity matrix and A is the 12 × 12 matrix:

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

Here, $A^T = A$

The binary linear code with generator matrix G is called the extended binary Golay code and will be denoted by G_{24}.

To encode, take the dot product of the message vector and G_{24}.

To decode, we use an algorithm involving the Syndrome matrix.

Here are the steps of the algorithm:

Step 1. Compute the syndrome $s = w\,H^T$.

Step 2. If $wt(s) \leq 3$ then **u = [s, 000000000000].**

Step 3. If $wt(s + b_i) \leq 2$ for some b_i of B then u $s + b_i, e_i$].

Step 4. Compute the second syndrome $s\,B$.

Step 5. If $wt(s\,B) \leq 3$ then **u = [0000000000(**$s\,B$ for some b_i of B

Step 6. I $wt(s\,B + b_i) \leq 2$ then u : $e_i, s\,B + b_i$ **].**

Step 7. If u is not yet determined then request retransmission.

Properties of the extended binary Golay code:

1) The length of G_{24} is 24 and its dimension is 12.

2) A parity-check matrix for G_{24} is the 12×24 matrix $H = [A \mid I_{12}]$.

Proof:

It is clear that H has the size required of a parity-check matrix and its rows are linearly independent. Hence it is enough to show that every row of H is orthogonal to every row of G, i.e., to show $GH^T = 0$. Indeed, we have

$$GH^t = [I_k|A]\begin{bmatrix} -A \\ I_{n-k} \end{bmatrix} = -I_k \cdot A + A \cdot I_{n-k} = -A + A = \mathbf{0}.$$

(n=12, k=2)

3) The code G_{24} is self-dual, i.e., $G^{\perp}_{24} = G_{24}$.

Proof:

The rows of G are orthogonal; i.e., if r_i and r_j are any two rows of G, then

$r_i \cdot r_j = 0$. This implies that $G_{24} \subset G^{\perp}_{24}$.

On the other hand, since both G_{24} and $G^{\perp}_{24}$ have dimension 12, we must have $G_{24} = G^{\perp}_{24}$.

4) Another parity-check matrix for G24 is the 12×24 matrix $H0 = [I_{12} \mid A]$ (= G).

Proof: A parity-check matrix of G24 is a generator matrix of $G^{\perp}_{24} = G_{24}$, and G is one such matrix.

5) Another generator matrix for G_{24} is the 12×24 matrix $G_0 = [A \mid I_{12}]$ (= H).

Proof: A generator matrix of G24 is a generator matrix of $G^{\perp}_{24} = G_{24}$, and G is one such matrix.

6) The weight of every codeword in G24 is a multiple of 4.

Proof:

Let v be a codeword in G_{24}.

We want to show that wt(v) is a multiple of 4.

Let r_i denote the i^{th} row of G.

First, suppose v is one of the rows of G.

Since the rows of G have weight 8 or 12, the weight of v is a multiple of 4.

Since v is a linear combination of the rows of G, let v be the sum :

$v = r_i + r_j$ of two different rows of G.

Since G24 is self-dual, the weight of v is divisible by 4.

By induction, we can prove that is the case for all the rows of the matrix.

Hence, every codeword has a weight that is a multiple of 4.

7) The code G_{24} has no codeword of weight 4, so the minimum distance of G24 is d = 8.

Proof:

The last row of G is a codeword of weight 8. This fact, together with statement 6), implies that d = 4 or 8. Suppose G_{24} contains a nonzero codeword v with wt(v) = 4. Write v as (v1,v2), where v1 is the vector (of length 12) made up of the first 12 coordinates of v, and v2 is the vector (also of length 12) made up of the last 12 coordinates of v. Then one of the following situations must occur:

Case 1: wt(v1) = 0 and wt(v2) = 4. This cannot possibly happen since, by looking at the generator matrix G, the only such word is 0, which is of weight 0.

Case 2: wt(v1) = 1 and wt(v2) = 3. In this case, again by looking at G, v must be one of the rows of G, which is again a contradiction.

Case 3: wt(v2) = 2 and wt(v2) = 2. Then v is the sum of two of the rows of G. It is easy to check that none of such sums would give wt(v2) = 2.

Case 4: wt(v1) = 3 and wt(v2) = 1. Since G_0 is a generator matrix, v must be one of the rows of G_0,which clearly gives a contradiction.

Case 5: wt(v1) = 4 and wt(v2) = 1. This case is similar to case 1, using G_0 instead of G.

Since we obtain contradictions in all these cases, d = 4 is impossible. Thus, d = 8.

8) The code G24 is an exactly three-error-correcting code.

Proof:

For a code to be exactly v-error correcting, it must satisfy:

$d(C) \geq 2v + 1$

but not:

$d(C) \geq 2(v+1) + 1$

Here d(C) = 8 and v = 3. The first condition is satisfied, but the second is not. Hence, it is exactly three-error correcting.

<u>Binary Golay Code:</u>

Let $\hat{G}$ be the 12 × 23 matrix $\hat{G} = [I12 \mid \hat{A}]$ where I_{12} is the 12 × 12 identity matrix and $\hat{A}$ is the 12 × 11 matrix obtained from the matrix A by deleting the last column of A. The binary linear code with generator matrix $\hat{G}$ is called the binary Golay code and will be denoted by G_{23}.

(Binary Golay code is essentially the code obtained from G_{24} by deleting the last digit of every codeword).

Properties of Binary Golay Code:

1)The length of G_{23} is 23 and its dimension is 12.

2 A parity-check matrix for G_{23} is the 11 × 23 matrix $\hat{H} = [\hat{A}^{t} | I_{11}]$.

3)The extended code of G_{23} is G_{24}.

4)The distance of G_{23} is d = 7.

5)The code G_{23} is a perfect exactly three-error-correcting code.

<u>Example:</u>

Suppose we receive the word 00000000000110000010111 over a noisy channel, we see immediately (because this word's Hamming weight is 7 and not 0,8,12,16 or 24) that some errors have occurred in transmission.

First we find the syndrome, wH^T = [0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,1,0,1,1,1].H^T
= [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]

Weight of this is 11, so we calculate Second Syndrome, s.B
= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Weight of this is 1, so u = $[e_i, sA+a_i]$

From u, we get [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1] as the final vector.

That means, the original message must have been 000000001001

Problem Statement

The aim of this study was to analyse the efficiency of two non-trivial perfect error correcting codes. The analysis was two fold, one in terms of the time taken to encode a given message and the second in terms of the time taken to decode and correct the errors within said message.

Approach and Implementation

We used Python to write encode and decode functions for each, and a final test function to see the results to make our qualitative comparison.

Hamming:

to_bin_array() : Converts an integer to a list of a given length containing it's binary representation.
get_parity_matrix(): Creates the parity matrix for a message of a given length.
encode(): Encodes the message by creating the generating matrix and multiplying it with the input vector.
decode(): Decodes the transmitted message by finding the position of the error and correcting it.

Golay:

encode() : dot product of the message vector with the Generator matrix
decode_extended_Golay() : decode using the syndrome method. Returns 11 bits to the decode function (omits the final parity bit)
decode() : adds the final parity bit and makes a call to decode_extended_Golay() for calculation. Returns the decoded version of the binary Golay-encoded message
find_weight() : calculates the weight of a given row vector

Summary

This paper outlined a detailed analysis of two non-trivial perfect linear error correcting codes, namly; the Hamming Code and the Golay Codes. These error correcting codes have been analysed both in terms of their mathematical implementation and their practical implementation; done through extensive mathematical proofs, detailed algorithms and functioning code.

Hamming and Golay Codes are two non-trivial perfect linear (block) error correcting codes. Hamming code is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. Golay code is a block code with a fixed length, size, and rate (unlike the Hamming code).

On analysis, we find that Golay encoding is much quicker than Hamming encoding. However, Hamming code corrects errors faster than Golay code.
We also find that Hamming codes require fewer redundant bits than Golay code, however it can only correct single bit errors, and detect two bit errors. Golay codes on the other hand

References

[1] Ling, San, and Chaoping Xing. 2004. Coding theory: a first course. Cambridge, UK: Cambridge University Press. (Chapters 1-5)

[2] https://en.wikipedia.org/wiki/Error_correction_code

[3] https://en.wikipedia.org/wiki/Linear_code

[4] https://blogs.ams.org/visualinsight/2015/12/01/golay-code/

[5] https://www.maplesoft.com/applications/view.aspx?SID=1757&view=html

[6] https://www.swissquant.com/wp-content/uploads/2017/09/Mathematical-Challenge-April-2013.pdf

[7] https://en.wikipedia.org/wiki/Hamming_code

[8] https://en.wikipedia.org/wiki/Error_detection_and_correction#Applications

[9] https://en.wikipedia.org/wiki/Binary_Golay_code

[10] http://giam.southernct.edu/DecodingGolay/introduction.html

[11]https://www.math.upenn.edu/~rbettiol/312files/ECC.pdf