1 Introduction to coding theory

1.1 Introduction

Coding theory is the study of methods for efficient and accurate transfer of information from one place to another. It is different to cryptography: we are no longer interested in secrecy, just accuracy and efficiency.

Coding theory has many uses: minimising noise on CD players, data transfer on phone lines or the internet, ethernet connections, data transfer from memory to CPU in a computer and space communication.

When transfering information from one place to another the information passes through a *channel*. The channel is the physical medium through which the information is transferred, for example the atmosphere or a phone line. Errors in the transferred information occur due to *noise* on the channel, that is, undesirable disturbances which may cause information received to differ from information transmitted. Noise can be caused by many things, for example, sunspots, lightning, poor typing, poor hearing.

Coding theory deals with the problem of *detecting* and *correcting* transmission errors caused by noise on the channel. The primary goals are to provide:

1. fast encoding of information;
2. easy transmission of encoded messages;
3. fast decoding of received message;
4. detection and correction of errors introduced in the channel;
5. maximum transfer of information per unit time.

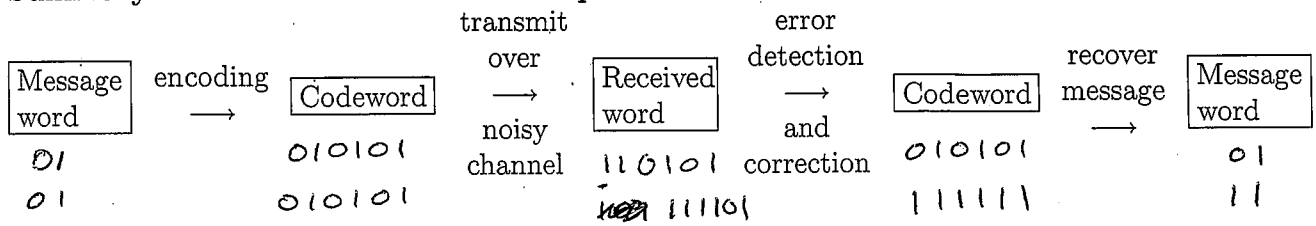
These goals are not necessarily compatible! Goal 4 is where we will spend most of our discussion.

Example 1.1 Consider normal conversation (codewords are english words, channel is atmosphere, encoder is speech, decoder is hearing). We have in-built error correction: if you received the message "apt natural, i have a gub", you would probably know what is meant. You can use the redundancy inherent in the message to infer its meaning.

Redundancy is a fundamental component of coding theory. We will be adding extra bits of information to each word before transmission in order to (hopefully) allow the effect of noise to be countered, and the correct word to be inferred. The challenge is to add as little extra information as possible, while still achieving the desired level of error detection and correction.

Example 1.2 *A 3-fold repetition code:* Suppose we have four message words: 00, 01, 10, 11, and we encode them by repeating them three times to get the four codewords 000000, 010101, 101010, 111111. After transmission, when we receive a word of length 6 we apply the decoding process of choosing the codeword which is closest to the received word. This system allows the detection of an error in up to two positions and the correction of an error in one position.

Summary of information transmission process:



Example 1.3 *The ISBN code:* Every recent book should have an International Standard Book Number (ISBN). This is a 10-digit codeword $x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}$ assigned by the publisher. The first digit indicates the language (0 for English) and the second and third digits indicate the publisher (for example, 19 stands for Oxford University Press). The next 6 digits are the book number assigned by the publisher. The final digit is a check digit, and is chosen so that the sum

$$\sum_{i=1}^{10} ix_i$$

is divisible by 11. (Note that the symbol X is used for the final digit to represent the number 10.) This system allows the detection of two types of errors: a single incorrect digit, and the transposition of two digits.

1.2 Basic definitions and assumptions

Definition 1.4 A *q-ary code* is a set of sequences of symbols where each symbol is chosen from a set of q distinct elements. The set of q distinct elements is called the *alphabet*. A sequence of symbols is called a *word* and a word that occurs in the code is called a *codeword*. The *length* of a word is the number of symbols in the word. A code in which each codeword has the same length, n say, is called a *block code of length n*. The number of codewords in a code C is denoted by $|C|$.

Example 1.5 The ISBN code is an 11-ary block code of length 10 based on the alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$.

Definition 1.6 2-ary codes are called *binary* codes, and are usually based on the alphabet $\{0, 1\}$. In this course, we focus on binary block codes, so the word code will refer to a binary block code unless otherwise indicated.

Definition 1.7 A binary codeword is transmitted by sending its digits, one at a time and in order, across a *binary channel*. A binary channel is *symmetric* if 0 and 1 are transmitted with equal accuracy; that is, the probability of receiving the correct digit is independent of whether the digit transmitted was 0 or 1. The *reliability* of a binary symmetric channel (BSC) is a real number $0 \leq p \leq 1$, where p is the probability that the digit sent is the digit received. The channel is *perfect* if $p = 1$.

We need to make a number of assumptions about the binary channel.

- We assume that any codeword of length n containing 0s and 1s will be received as a word of length n containing 0s and 1s, although not necessarily the same as the original codeword. We also assume that it is easy to identify the beginning of the first word received, and hence that we can identify each received word.
- We assume that noise on a channel is scattered randomly, rather than in bursts. Thus, the probability of any one digit being altered by noise is equal to the probability of any other digit being altered. (This is perhaps not reasonable, but we will relax this later on.)
- We assume that no channel is perfect.
- If p is the probability that the digit received is the same as that sent, then $1-p$ is the probability that the digit received **is not** the same as the digit sent. If any channel has $p < 0.5$, this can be converted to a channel with $0.5 \leq p \leq 1$ (by inverting each digit), so we will assume that we are communicating via a symmetric binary channel with $0.5 \leq p < 1$.

Definition 1.8 Let v be the codeword of length n transmitted, w be the word received and assume communication is over a BSC with reliability p and with randomly scattered noise. Let $\Phi_p(v, w)$ be the probability that if v is transmitted then w is received. If v and w disagree in d positions then we have

$$\Phi_p(v, w) = p^{n-d}(1-p)^d.$$

Exercise 1.9 Let C be a code of length 6, and suppose that we are transmitting codewords over a BSC with reliability $p = 0.95$ and with randomly scattered noise.

1. For any codeword $v \in C$, what is the probability that v is received correctly? $(0.95)^6 \approx 0.735$
2. Let $v = 101010 \in C$ and $x = 001010$. What is $\Phi_p(v, x)$? $(0.95)^5(0.05)^1 \approx 0.0387$
3. Let $v = 101010 \in C$ and $w = 011010$. What is $\Phi_p(v, w)$? $(0.95)^4(0.05)^2 \approx 0.002$
4. For any codeword $v \in C$, what is the probability that a word is received which differs from v in one position (so one error has occurred)? $\binom{6}{1}(0.95)^5(0.05)^1$
5. For any codeword $v \in C$, what is the probability that a word is received which differs from v in two positions (so two errors have occurred)? $\binom{6}{2}(0.95)^4(0.05)^2$
6. For any codeword $v \in C$, what is the probability that a word is received which differs from v in two or more positions?

$$1 - \binom{6}{1}(0.95)^5(0.05)^1 - \binom{6}{0}(0.95)^6$$

or $\binom{6}{2}(0.95)^4(0.05)^2 + \binom{6}{3}(0.95)^3(0.05)^3 + \binom{6}{4} \ldots + \binom{6}{6} \ldots$

If the reliability of the channel was only $p = 0.51$, then the answers to questions 1 – 3 above would be 0.0176, 0.0169 and 0.0162, respectively. It is vital to have a reliable channel in order to have any chance of receiving transmitted information correctly. In practice, channel reliability is usually higher than 0.95.

1.3 Introduction to error detection

Suppose a word is received, and this word is not a codeword. Then we have *detected* that at least one error has occurred in transmission.

Example 1.10 Let $C_1 = \{00, 01, 10, 11\}$. Then C_1 cannot detect any errors in transmission.

Example 1.11 Let C_2 be a new code formed by repeating each codeword of C_1, so

$$C_2 = \{0000, 0101, 1010, 1111\}.$$

If a single error occurs in any transmitted codeword w, then the received word will not be a codeword. This allows detection of any single error. If two errors occur in any transmitted codeword w, then the received word may or may not be a codeword. So some sets of two errors will be detected, but not all sets of two errors.

0101 -> 1111 don't detect
0101 -> 1001 will detect.

We call C_2 a *repetition* code: in fact, it is a 2-*fold repetition code*, denoted $Rep(2)$. An n-fold repetition code is formed by taking the 2^k words of length k and forming the codewords of length kn by writing down, n times, each word of length k.

Example 1.12 Let C_3 be a new code formed from C_1 by adding a third digit to each codeword so that the number of 1s in each codeword is even, so $C_3 = \{000, 011, 101, 110\}$. The added digit is called a *parity-check* digit, and it enables detection of any single error. This code will not detect any set of two errors in a transmitted codeword.

Exercise 1.13 Consider a communication channel with reliability $p = 1 - 10^{-8}$. Consider the code C consisting of all 2^{11} words of length 11 (so there are no check digits). Suppose that digits are transmitted at 10^7 digits per second. On average, approximately how many words are (undetectedly) transmitted incorrectly per minute?

Prob of receiving an incorrect word $= 1 - p^{11}$

$= 1 - (1 - 10^{-8})^{11}$

$\approx 1.1 \times 10^{-7}$

$$\frac{10^7 \text{ digits}}{\text{second}} \times \frac{60 \text{ seconds}}{1 \text{ minute}} \times \frac{1 \text{ word}}{11 \text{ digits}} = \frac{60}{11} \times 10^7 \text{ words per minute}$$

incorrect words per minute

$$\frac{60}{11} \times 10^7 \times 1.1 \times 10^{-7} \approx 6 \text{ words per minute are incorrect.}$$

Now let D be the code obtained from the code C in Example 1.13 by adding a parity-check digit to each codeword of C, so that the number of 1s in each transmitted word is even. Using the same reliability and rate of transmission, we will determine how many words are (undetectedly) transmitted incorrectly per minute if the code D is used. The code D will detect a single error in a transmitted word, so the probability of an incorrect word being received and not detected is

$$1 - P(0 \text{ errors}) - P(1 \text{ error}) = 1 - \binom{12}{0} p^{12}(1-p)^0 - \binom{12}{1} p^{11}(1-p)^1 \approx 6.6 \times 10^{-15}.$$

Words are transmitted at a rate of

$$\frac{10^7 \text{ digits}}{1 \text{ second}} \times \frac{1 \text{ word}}{12 \text{ digits}} \times \frac{60 \text{ seconds}}{1 \text{ minute}} = 5 \times 10^7 \text{ words per minute.}$$

Thus approximately 3.3×10^{-7} incorrect words are undetectedly transmitted per minute. That is approximately 1 word in 6 years.

By adding only a small amount of extra information (redundancy) we drastically reduced the number of incorrect words that slip through without detection.

1.4 Information rate of a code

In the previous subsection, we created two new codes, C_2 and C_3, by adding extra bits of information (redundancy) to the codewords of C_1, thus enabling error detection. In each case we added different amounts of information, but each code can still only detect a *single* error. In some sense, the code C_3 may be more efficient than the code C_2.

Clearly, by adding extra bits of information to the words, we can improve error detection. Of course, as check digits are added, more bits must be transmitted for each codeword, thereby increasing transmission time.

Definition 1.14 Many codes are obtained by taking the 2^k words of length k and adding $n-k$ *check bits* to each word, thus giving codewords of length n. A code of length n, with 2^k codewords is called an (n, k) code. The number k is the *dimension* of the code, and we say that such a code has k *message bits*.

Definition 1.15 If C is any code of length n then the *information rate* or *rate* of C is given by

$$\frac{1}{n} \log_2 |C|.$$

Hence if C is an (n, k) code (so $|C| = 2^k$) then the information rate of C is k/n.

Exercise 1.16 Compare the information rates of the codes C and D from Exercise 1.13 and the discussion following it.

Rate of code $C = 1$ Rate of code $D = \frac{11}{12}$

Thus, for a (reasonably) small reduction in efficiency/information rate, it is possible to incorporate extra information, allowing detection of a single error.

1.5 Introduction to error correction

What can be done if the existence of an error is detected? Requesting the retransmission of a message has a significant cost: we need to interrupt and delay transmission. It would be much better if we could not only detect the *existence* of an error, but could also *locate* it. If we can locate where the error occurred, then we can *correct* it by inverting the received bit.

Example 1.17 Let $Rep(3)$ be the 3-fold repetition code formed by writing down each word of length 2 three times, so $Rep(3) = \{000000, 010101, 101010, 111111\}$. If a single error occurs in any digit of any transmitted codeword v, then the received word w will not be a codeword. This allows detection of a single error. Moreover, the received word must have originated from one of the codewords. It differs from one codeword (v) in one place, and from the other codewords in more than one place. Hence the *most likely* word sent was v, so it makes sense to decode the received word as v. This is an example of error *correction.*

Exercise 1.18 Consider the code $Rep(3)$ from Example 1.17. If the word 100010 is received, what is the most likely codeword to have been sent? 100010 most likely codeword 101010

The decoding method used in the previous example makes intuitive sense as a decoding mechanism: correct any received word to the codeword which differs from the received word in as few bit places as possible. We can formally prove that this is valid. Recall that $\Phi_p(v, w)$ is the probability that the word w is received if the codeword v is transmitted over a BSC with reliability p. In practice, we know the word received w, but we do not know the codeword transmitted v. However, we know all of the codewords, so can calculate $\Phi_p(u, w)$ for each codeword $u \in C$. Clearly, we want to choose the *most-likely* transmitted word, which means we choose the codeword v for which

$$\Phi_p(v, w) = \max\{\Phi_p(u, w) | u \in C\}.$$

We can choose such a codeword via the following theorem:

Theorem 1.19 *Suppose communication is via a BSC with reliability p, $0.5 < p < 1$. Let v_1 and v_2 be codewords of length n, and w a word of length n. Suppose that v_1 and w disagree in d_1 positions, and that v_2 and w disagree in d_2 positions. Then*

$$\Phi_p(v_1, w) \leq \Phi_p(v_2, w) \text{ if and only if } d_1 \geq d_2.$$

Proof: We have

$$\begin{aligned} & \Phi_p(v_1, w) \leq \Phi_p(v_2, w) \\ \leftrightarrow\ & p^{n-d_1}(1-p)^{d_1} \leq p^{n-d_2}(1-p)^{d_2} \\ \leftrightarrow\ & \left(\frac{p}{1-p}\right)^{d_2-d_1} \leq 1 \\ \leftrightarrow\ & d_2 \leq d_1 \end{aligned}$$

(since $\frac{p}{1-p} > 1$). □

if $prob(v_1, w) \leq prob(v_2, w)$ then v_2 is closer to w than v_1 is

Thus, as we wish to maximise $\Phi_p(u, w)$, we correct the received word w to the codeword v which differs from w in as few positions as possible.

Exercise 1.20 Suppose that $w = 0010110$ is received over a BSC with $p = 0.9$. Which of the following codewords is most likely to have been sent?

$$1001011, 1111100, 0001110, 0011001, 1101001.$$

What would have been the case if the channel instead had reliability $p = 0.51$?

0001110

↑ same answer but it is much less likely to be correct.

1.6 Weights and distances

We need an efficient way of finding which codeword is closest to a received word. If there are many codewords, it's not practical to check every received word against every possible codeword. (For example, the code used on the Voyager Mission had $2^{12} = 4096$ codewords.)

Recall that K^n consists of all the binary vectors (words) of length n.

Exercise 1.21 If u, v, w are words in K^n, show that

1. $v + w = \mathbf{0}$ iff $v = w$;
2. if v is transmitted over a BSC and w is received, then $u = v + w$ will be a word containing a 1 in exactly those places in which v and w differ.

(1) In mod 2 $1+1=0$, $0+0=0$, $1+0=1$ and $0+1=1$

Let $\underset{\sim}{v} = v_1 v_2 \cdots v_n$ and $\underset{\sim}{w} = w_1 w_2 \cdots w_n$

$v_i + w_i = 0$ if and only if $v_i = w_i$ so $\underset{\sim}{v} + \underset{\sim}{w} = \underset{\sim}{0}$ if and only if each $v_i = w_i$.

(2) $v_i + w_i$ will be 0 when $v_i = w_i$ and will be 1 when $v_i \neq w_i$ so we will get a 1 in exactly the places in which $\underset{\sim}{v}$ and $\underset{\sim}{w}$ differ.

Definition 1.22 Given two words v and w in K^n, the corresponding *error pattern* or *error* is defined by $u = v + w$, and is 1 in exactly those places in which v and w differ.

Definition 1.23 Let $v \in K^n$ be a word of length n. Then the *weight* or *Hamming weight* of v is the number of occurrences of the digit 1 in v. We denote the weight of a word v as $wt(v)$.

Definition 1.24 The *Hamming distance* $d(u, v)$ between two words $u, v \in K^n$ is the number of places in which their bits differ.

$d(u,v) = wt(u+v)$

Exercise 1.25 Let $x = 111001$, $y = 001111$ and $z = 101010$. Find $wt(x)$, $d(x, y)$ and $d(y, z)$.

$wt(x) = 4$
$d(x, y) = wt(x+y) = wt(110110) = 4$
$d(y, z) = wt(y+z) = wt(100101) = 3$

Hamming distance satisfies the properties of a *metric*. That is, if $x, y, z \in K^n$, then

1. $d(x, y) \geq 0$
2. $d(x, y) = 0$ iff $x = y$
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

Note that if v and w are codewords and u is the error pattern $u = v+w$, then we have $d(v, w) = wt(u)$. That is, $d(v, w) = wt(v + w)$.

If $u = v + w$, then the probability formula from Theorem 1.19 can be rewritten as

$$\Phi_p(v, w) = p^{n-wt(u)}(1 - p)^{wt(u)}.$$

We refer to $\Phi_p(v, w)$ as the *probability of the error pattern* $u = v + w$.

1.7 Maximum Likelihood Decoding

Now we are ready to describe more formally how decoding is done in general. There are two commonly used approaches to decoding.

Definition 1.26 *Complete Maximum Likelihood Decoding* or *CMLD*
Let C be the set of codewords, $v \in C$ be the codeword transmitted, and w the word received. If there is a unique codeword $v \in C$ for which $d(v, w) < d(v_1, w)$ for all $v_1 \in C, v_1 \neq v$, then we decode w as v. If there are two or more codewords which are all closest to w, then we arbitrarily select one of the equally closest codewords and decode w as that arbitrary choice.

Definition 1.27 *Incomplete Maximum Likelihood Decoding* or *IMLD*
Again, if there is a unique word v in C closest to w then we decode w as v. However, if there are two or more codewords which are all equally close to w, then we request retransmission.

Unless stated otherwise, we will always assume that IMLD is being used.

Note that with Maximum Likelihood Decoding, we are always going to select the codeword which is *closest* to the received word. This is not necessarily the same as selecting the *correct* or *transmitted*

codeword: it may be the case that so many errors occurred in transmission that the codeword closest to the received word is not the same as the codeword which was transmitted.

One of our aims is to ensure that decoding to the *closest* codeword will (almost) always produce the *correct* codeword.

Using IMLD, the codeword $v_1 \in C$ closest to the received word w is the codeword for which $d(v_1, w)$ is least. By Theorem 1.19, this codeword v_1 has the largest probability $\Phi_p(v_1, w)$ of being the codeword transmitted. Since $d(v_1, w) = wt(v_1 + w)$, we can restate the result of Theorem 1.19 as

$$\Phi_p(v_1, w) \leq \Phi_p(v_2, w) \text{ iff } wt(v_1 + w) \geq wt(v_2 + w).$$

That is, *the most likely codeword sent is the one with the error pattern of smallest weight.*

Thus given a received word w, the decoding strategy for IMLD is to examine the error patterns $u = v + w$ for all codewords $v \in C$ and decode w as the codeword v^* which gives the error pattern u^* of smallest weight.

Example 1.28 If $C = \{0000, 1010, 0111\}$, construct an IMLD table showing, for each possible received word w, to which codeword w will be decoded. Remember that retransmission will be requested in the event that there is no unique codeword which minimises the weight of the error pattern.

received word	error patterns			most likely codeword
w	$0000 + w$	$1010 + w$	$0111 + w$	v
0000	0000	1010	0111	0000
0001	0001	1011	0110	0000
0010	0010	1000	0101	−−
0011	0011	1001	0100	0111
0100	0100	1110	0011	0000
0101	0101	1111	0010	0111
0110	0110	1100	0001	0111
0111	0111	1101	0000	0111
1000	1000	0010	1111	−−
1001	1001	0011	1110	−−
1010	1010	0000	1101	1010
1011	1011	0001	1100	1010
1100	1100	0110	1011	−−
1101	1101	0111	1010	0111
1110	1110	0100	1001	1010
1111	1111	0101	1000	0111

When establishing a BSC and a code, it is necessary to choose the value of n (length of the codewords) and the actual codewords C. Some choices of n and C are better than others. There are many criteria that are used in the choices of n and C. For now we would like to choose codewords C such that IMLD works reasonably well. To do that we need to determine, for each codeword v, the probability that IMLD will correctly conclude that v was transmitted.

Given n and C, we can calculate the probability $\Theta_p(C, v)$ that if $v \in C$ is transmitted over a BSC with reliability p then IMLD correctly concludes that v was sent.

Definition 1.29 *Calculating the reliability of IMLD*
Assume that a codeword $v \in C$ is transmitted over a BSC with reliability p. Let

$$L(v) = \{x \mid x \in K^n \text{ and } x \text{ is closer to } v \text{ than to any other codeword in } C\}.$$

Then $\Theta_p(C, v)$ is the sum of all the probabilities $\Phi_p(v, w)$ as w ranges over $L(v)$. That is,

$$\Theta_p(C, v) = \sum_{w \in L(v)} \Phi_p(v, w).$$

$L(v)$ is precisely the set of words in K^n for which, if received, IMLD will conclude that v was sent. We can find $L(v)$ from an IMLD table, such as the one given above, by comparing the received words with the most likely codewords.
Note that the definition of Θ_p ignores the effect of retransmission when decoding is ambiguous (so the received word might be correctly decoded on second transmission), but Θ_p is still a reasonable lower bound on the probability of correct decoding.

Example 1.30 Suppose $p = 0.9$, $n = 4$ and $C = \{0000, 1010, 0111\}$ (as in Example 1.28). Compute $\Theta_p(C, 0000)$.

For $v = 0000$, $L(v) = \{0000, 0100, 0001\}$. Thus

$$\begin{aligned} \Theta_p(C, 0000) &= \Phi_p(0000, 0000) + \Phi_p(0000, 0100) + \Phi_p(0000, 0001) \\ &= p^4 + p^3(1-p) + p^3(1-p) \\ &= 0.8019 \end{aligned}$$

Exercise 1.31 Let $p = 0.9$, $n = 4$ and $C = \{0000, 1010, 0111\}$ (as in Examples 1.28 and 1.30. Compute $\Theta_p(C, 1010)$ and $\Theta_p(C, 0111)$.

$\Theta_p(C, 1010)$:

$L(1010) = \{1010, 1011, 1110\}$

$\Theta_p(C, 1010) = p^4 + 2p^3(1-p)$

$= 0.8019$

$L(0111) = \{0011, 0101, 0110, 0111, 1101, 1111\}$

$\Theta_p(C, 0111) = p^4 + 4p^3(1-p) + p^2(1-p)^2$

$= 0.9511$

We can see that for $C = \{0000, 1010, 0111\}$, IMLD works reasonably well if 0111 is transmitted, but performs poorly if 0000 or 1010 is transmitted. Thus we see that C is not a very good choice as a code.

1.8 Error detection and correction

Now we can formalise the definition of error detection.

Definition 1.32 We say that a code C *detects* the error pattern u if and only if $v+u$ is not a codeword, for every $v \in C$.

Exercise 1.33 Let $C = \{001, 101, 110\}$. Show that C detects the error pattern $u_1 = 010$, but does not detect the error pattern $u_2 = 100$.

$001 + 010 = 011 \notin C$
$101 + 010 = 111 \notin C$
$110 + 010 = 100 \notin C$
So C detects the error pattern 010.

$001 + 100 = 101 \in C$
So C does not detect the error pattern 100.

A good method of determining which error patterns a code can detect is to first determine which error patterns it *cannot* detect. Given a code C and any pair of codewords v and w, if $e = v + w$ then C cannot detect the error pattern e (as $v+e = w$ which is a codeword). Thus the set of all error patterns which cannot be detected by C is the set of all words that can be written as the sum of two codewords.

Exercise 1.34 Find all error patterns which can be detected by $C = \{1000, 0100, 1111\}$.

Cannot detect $1000 + 0100 = 1100$
$1000 + 1111 = 0111$
$0100 + 1111 = 1011$

Can detect any error pattern in $K^4 - \{1100, 0111, 1011\}$.

For certain codes, we can calculate some of the error patterns which the code can detect, without needing to go through any of the above calculations. We make use of the concept of the *distance* between two codewords, defined in Definition 1.24, and define the concept of the *distance of a code.*

Definition 1.35 The *distance* (or *minimum distance*) of a code is the smallest distance between any pair of distinct codewords. That is, we define δ to be the *distance* of a code C if

$$\delta = \min_{v,w \in C, v \neq w} d(v, w).$$

We know that $d(v,w) = wt(v+w)$, so δ is the smallest value of $wt(v+w)$ as v, w, $v \neq w$ range over all possible pairs of codewords.

An (n,k) code with minimum distance δ will sometimes be written as a (n,k,δ) code.

Exercise 1.36 Find the distance of the code $C = \{0000, 1010, 0111\}$.

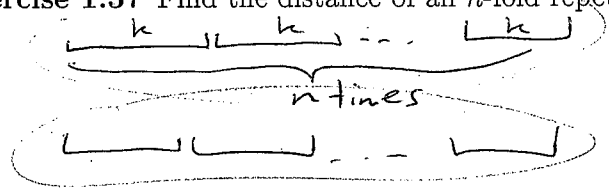
$0000 + 1010 = 1010$

$0000 + 0111 = 0111$

$1010 + 0111 = 1101$

$\delta_C = 2$

Exercise 1.37 Find the distance of an n-fold repetition code.



000 000 000 ... 000

100 100 100 ... 100

$\delta = n$

We now have a very important theorem, which makes direct use of the definition of distance given above.

Theorem 1.38 *Let C be a code with minimum distance δ. Then C will detect all non-zero error patterns of weight less than or equal to $\delta - 1$. Moreover, there is at least one error pattern of weight δ which C will not detect.*

Proof: Let u be a non-zero error pattern with $wt(u) \leq \delta - 1$, and let $v \in C$. Then $d(v, v+u) = wt(v + (v+u)) = wt(u) < \delta$. Since C has distance δ, $v + u \notin C$. Therefore C detects u.
From the definition of δ, there exist codewords $v, w \in C$ with $d(v,w) = \delta$. Consider the error pattern $u = v + w$. Now $w = v + u \in C$, so C will not detect the error pattern u of weight δ. □

Note that a code C with minimum distance δ may possibly detect *some* error patterns of weight δ or more, but it does not detect *all* error patterns of weight δ.

Exercise 1.39 Show that the code defined in Exercise 1.36 detects the error pattern 1111 of weight 4, but find an error pattern of weight 2 which C does not detect.

$0000 + 1111 = 1111 \notin C$ so C detects the error pattern 1111

$1010 + 1111 = 0101 \notin C$ but C does detect the error

$0111 + 1111 = 1000 \notin C$ pattern 1010.

Definition 1.40 A code C is said to be x *error detecting* if it detects **all** error patterns of weight at most x, and does not detect at least one error pattern of weight $x+1$. By Theorem 1.38, if C has distance δ then C is $\delta - 1$ error detecting.

Equivalently, for a code C to be e error detecting, it must have distance $e+1$.

Exercise 1.41 Let $C = \{0000, 1010, 0111\}$.

1. What is the distance of C? $d = 2$
2. C is x-error detecting. What is x? $x = 1$
3. Find all error patterns that C does detect, and hence show that 1010 is the only error pattern of weight 2 that C does not detect.

From Exercise 1.36, C cannot detect $\{1010, 0111, 1101\}$

So C can detect $K^4 - \{1010, 0111, 1101\}$

Now we can formalise the definition of error correction.

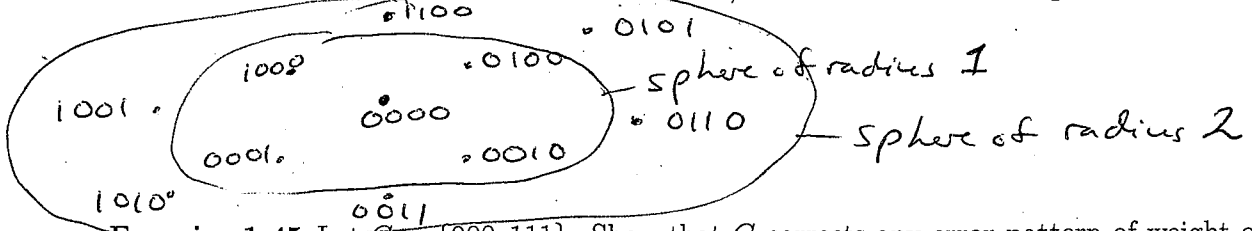
Definition 1.42 A code C *corrects an error pattern* u if, for all $v \in C$, $v + u$ is closer to v than to any other word in C.

This is equivalent to saying that C corrects the error pattern u if adding u to any codeword v results in a word which is still closer to v than to any other codeword.

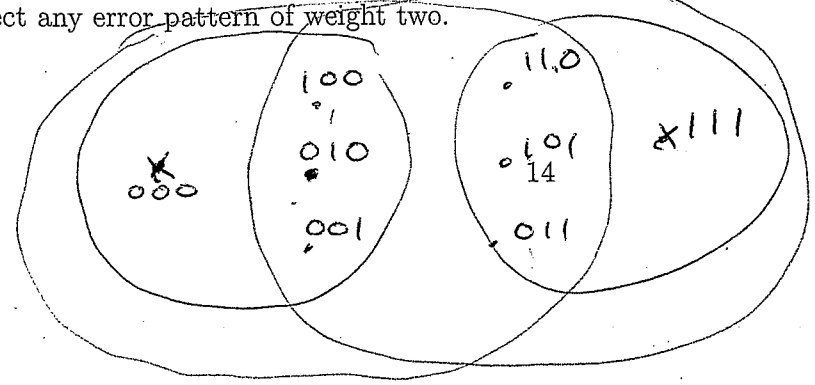
Definition 1.43 A code C is said to be x *error correcting* if it corrects all error patterns of weight at most x, and does not correct at least one error pattern of weight $x + 1$.

Given a codeword v we can think of a "sphere" in n dimensions of radius x centred on the codeword v by saying that another word w falls within this sphere iff w is within distance x of codeword v. Intuitively, C is x error correcting if it possible to take each codeword in C, draw a "sphere of radius x" over each codeword, and have no two spheres intersect. Then any received word which falls within the sphere of a codeword will be corrected unambiguously to the codeword on which the sphere is based.

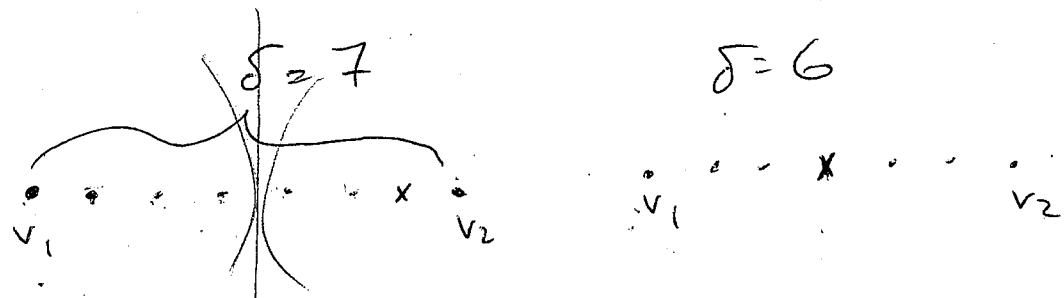
Exercise 1.44 If C is a ~~code and $v = 0000 \in C$~~, list all words within a sphere of radius 2 of v.



Exercise 1.45 Let $C = \{000, 111\}$. Show that C corrects any error pattern of weight one, but does not correct any error pattern of weight two.



Because spheres of radius 1 do not overlap, any error pattern of weight 1 can be corrected.



The process of IMLD picks the "most-likely" codeword. When we relate error correction and the distance of a code, we need to ensure that the most-likely codeword is the correct codeword, and not just the closest codeword.

Let v_1, v_2 be codewords with $d(v_1, v_2) = \delta$. Clearly, if v_1 is transmitted and errors occur in $\delta - 1$ of the places in which v_1 and v_2 differ, the received word will be the same as if v_2 had been transmitted and a single error had occurred in the other place in which v_1 and v_2 differ. Thus $\delta - 1$ errors in v_1 can be equivalent to 1 error in v_2. Similarly, $\delta - 2$ errors in v_1 can give the same received word as 2 errors in v_2, and so on.

Suppose that we are using a code C with distance δ and a codeword v is transmitted and the word w is received where $d(v, w) = \delta - 1$. Then we can *detect* that up to $\delta - 1$ errors have occured. However, care must be taken with error correction since $\delta - 1$ errors in v may be indistinguishable from just 1 error in a different codeword and so in choosing the closest codeword to the received word w, the process of IMLD may return the incorrect codeword.

Example 1.46 Let $C = \{000, 111\}$, so C has distance 3 and hence can detect all error patterns of weight at most 2. If the word $w = 010$ is received, it is more likely (and error correction will assume) that the error pattern which occurred was 010 (of weight 1, with transmitted word 000), rather than an error pattern 101 (of weight 2, with transmitted word 111).

We now state a fundamental theorem which forms the basis for error correcting codes.

Theorem 1.47 *Let C be a code with minimum distance δ.*

If δ is odd, then C can correct all error patterns with weight less than or equal to $\dfrac{\delta - 1}{2}$.

If δ is even, then C can correct all error patterns with weight less than or equal to $\dfrac{\delta - 2}{2}$.

Theorem 1.47 can be justified intuitively. If C has distance δ, then any two codewords are at least distance δ apart. If δ is odd and up to $(\delta - 1)/2$ errors occur, or if δ is even and up to $\delta/2$ errors occur, then the received word will not be a codeword, so it is clearly possible to *detect* that this number of errors has occurred. In most of these cases, it is also possible to *correct* the error, by selecting the unique codeword which is closest to the received word. However, if δ is even and $\delta/2$ errors have occurred, then it is possible for the received word to be equidistant from two distinct codewords. Thus it may not be possible to unambiguously select the *closest* codeword, so error correction may not work.

Using $\lfloor x \rfloor$ to denote the integer part of x, we see that a code C of distance δ can correct up to $\left\lfloor \dfrac{\delta - 1}{2} \right\rfloor$ errors. We note that there is at least one error pattern of weight $1 + \left\lfloor \dfrac{\delta - 1}{2} \right\rfloor$ which C does not correct. A code of distance δ *may* correct *some* error patterns of weight larger than that specified in Theorem 1.47. However, it will not correct *all* such error patterns.

Exercise 1.48 Assume that the information to be transmitted consists of all possible strings of length 3. Label the message bits x_1, x_2 and x_3, and let C contain codewords of length $n = 6$ formed by appending three extra digits x_4, x_5 and x_6, so that each of the following sums are even:

$$x_2 + x_3 + x_4, \; x_1 + x_3 + x_5, \; x_1 + x_2 + x_6.$$

1. List the codewords of C.
2. What is the distance of C?
3. How many errors can C detect and correct?
4. What is the information rate of C?

000 000
100 011
010 101
001 110
110 110
101 101
011 011
111 000

$\delta = 3$ detect 2 rate: $\frac{1}{2}$
correct 1
consider $x_4 x_5 x_6$

Exercise 1.49 Repeat the previous example, but instead form a code D by repeating each of the three message bits three times.

000 000 000
100 100 100
010 010 010
001 001 001
110 110 110
101 101 101
011 011 011
111 111 111

$\delta = 3$ detect 2 correct 1 rate: $\frac{1}{3}$

Exercise 1.50 Compare the rates of codes C and D from the two previous examples.

C is more efficient

Later, we will see how to construct a family of codes called the *Hamming codes*. We will see that the Hamming code of length 7 is 2 error detecting and 1 error correcting, but has information rate 4/7.

2 Linear codes I

The essential goal of coding theory is to find codes which transmit information at reasonable rates, yet also detect and correct most transmission errors. In this section we discuss a broad class of codes which provide these features, based heavily on linear algebra.

2.1 Introduction to linear codes

Definition 2.1 A **linear code** is a code in which the sum (mod 2) of any two codewords is also a codeword. That is, C is linear iff for any pair of codewords $v, w \in C$, we also have $v + w \in C$.

Almost every code we consider for the remainder of this course will be linear.

Exercise 2.2 Show that $C_1 = \{0000, 0101, 1010, 1111\}$ is linear, but that $C_2 = \{0000, 1001, 1010, 0011, 1111\}$ is not linear.

$0000 + \underset{\sim}{v} = \underset{\sim}{v}$

$0101 + 1010 = 1111$

C_1 is linear

C_2: $1111 + 0011 = 1100 \notin C_2$

so C_2 is not linear.

Exercise 2.3 Explain why any linear code must contain the zero word.

$\underset{\sim}{v} + \underset{\sim}{v} = \underset{\sim}{0}$

Exercise 2.4 Five of the eight codewords of a linear code are

$$0001111, 0110101, 1010011, 1011100, 1100110.$$

Find the remaining three codewords.

00000000

$0111010 = x + y$

$x + z = 1101001$

One of the advantages of a linear code is that its distance is easy to find.

Theorem 2.5 *For any linear code C, the distance of C is the weight of the nonzero codeword of smallest weight.*

$d(\underset{\sim}{0}, \underset{\sim}{w}) = wt(\underset{\sim}{w}) \geq \delta$

Proof: Let C be a code of distance δ, and let w be the nonzero codeword of smallest weight. Certainly, $\delta \leq wt(w)$. Assume that there are two codewords $v_1, v_2 \in C$ such that $\delta = d(v_1, v_2) = d < wt(w)$. As C is linear, $v = v_1 + v_2$ must be a codeword, of weight $d(v_1, v_2) < wt(w)$. But this contradicts the assumption that w is the nonzero codeword of smallest weight. □

Exercise 2.6 Find the distance of the linear code $C = \{0000, 1100, 0011, 1111\}$.

$\delta = 2$

It is easy to find the distance of a linear code. Other advantages of linear codes include:

1. For linear codes, there is a procedure for IMLD which is simpler and faster than we have seen so far (for some linear codes, there are very simple decoding algorithms).
2. Encoding using a linear code is faster and requires less storage space than for arbitrary non-linear codes.
3. The probabilities $\Theta_p(C, v)$ are straightforward to calculate for a linear code.
4. It is easy to describe the set of error patterns that a linear code will detect.
5. It is much easier to describe the set of error patterns a linear code will correct than it is for arbitrary non-linear codes.

Since a subset $U \subseteq K^n$ is a subspace of K^n iff U is closed under addition, we conclude that C is a linear code iff C is subspace of K^n.
Thus, for any subset S of K^n, the span of S is a linear code, $C = \langle S \rangle$.
The *dimension of a linear code* is the dimension of the corresponding subspace of K^n. Similarly, a *basis for a linear code* is a basis for the corresponding subspace of K^n.
If a linear code C has dimension k and if $B = \{v_1, v_2, \ldots, v_k\}$ is a basis for C, then each codeword w in C can be written as

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k$$

for a unique choice of digits $\alpha_1, \alpha_2, \ldots, \alpha_k$. Noting that each α_i is 0 or 1, for $1 \leq i \leq k$, there are 2^k distinct choices for $\alpha_1, \alpha_2, \ldots, \alpha_k$.
We thus have the following very important theorem:

Theorem 2.7 *A linear code of dimension k contains precisely 2^k codewords.*

Thus, using the notation introduced earlier in the course, a linear code with length n, dimension k and distance δ is an (n, k) linear code of distance δ, or equivalently, an (n, k, δ) linear code. Such a code has information rate k/n.

2.2 Dual codes

We now see how to derive a new code from a given linear code, using the orthogonal complement.

Definition 2.8 For $S \subseteq K^n$, if $C = \langle S \rangle$, then we write $C^\perp = S^\perp$ and call $C^\perp$ the *dual code* of C.

Theorem 2.9 *Let $C = \langle S \rangle$ be the linear code generated by a subset S of K^n. If the dimension of C is k_1 and the dimension of $C^\perp$ is k_2 then we must have $k_1 + k_2 = n$.*

Exercise 2.10 Suppose that C is a $(9, 4)$ linear code. How many codewords are in C? How many codewords are in $C^\perp$?

$|C| = 2^k = 16$

dim of $C = 4$

dim of $C^\perp = 5$

$|C^\perp| = 32$.

$16 + 32 \neq 512$

$C \cap C^\perp$ is not necessarily $\emptyset$.

2.3 Bases for codes

In this section we develop methods for finding bases for a linear code $C = \langle S \rangle$ and its dual $C^{\perp}$.

Algorithm 2.11 Algorithm for finding a basis for $C = \langle S \rangle$.
Let S be a nonempty subset of K^n. Form the matrix $\mathbf{A}$ whose **rows** are the words in S. Use EROS to find a REF of $\mathbf{A}$. Then the nonzero rows of the REF of $\mathbf{A}$ form a basis for $C = \langle S \rangle$.

Algorithm 2.11 works because the rows of $\mathbf{A}$ generate C, and EROS simply interchange codewords (rows) or replace one codeword (row) with the sum of two rows (another codeword) giving a new set of codewords which still generates C. Clearly the nonzero rows in a matrix in REF are linearly independent. Note that Algorithm 2.11 does not produce a unique basis for $C = \langle S \rangle$, and there is no guarantee that the words in the basis occur in the given set S.

Exercise 2.12 Use Algorithm 2.11 to find a basis for the linear code $C = \langle S \rangle$ where $S = \{11101, 10110, 01011, 11010\}$.

$$\begin{pmatrix} 1&1&1&0&1 \\ 1&0&1&1&0 \\ 0&1&0&1&1 \\ 1&1&0&1&0 \end{pmatrix} \rightarrow \begin{pmatrix} 1&1&1&0&1 \\ 0&1&0&1&1 \\ 0&1&0&1&1 \\ 0&0&1&1&1 \end{pmatrix} \rightarrow \begin{pmatrix} 1&1&1&0&1 \\ 0&1&0&1&1 \\ 0&0&1&1&1 \\ 0&0&0&0&0 \end{pmatrix} \text{REF}$$

Basis is $\{11101, 01011, 00111\}$

C has 8 codewords.

Now we give an algorithm for finding a basis for the dual code $C^{\perp}$. This algorithm incorporates Algorithm 2.11, so it also gives a basis for C.

Algorithm 2.13 Algorithm for finding bases for C and $C^{\perp}$.
Let S be a nonempty subset of K^n. Form the matrix $\mathbf{A}$ whose rows are the words in S. Use EROS to place $\mathbf{A}$ in RREF. Let $\mathbf{G}$ be the $k \times n$ matrix consisting of all the nonzero rows of the RREF. Then the rows of $\mathbf{G}$ form a basis for C. Let $\mathbf{X}$ be the $k \times (n-k)$ matrix obtained from $\mathbf{G}$ by deleting the leading columns of $\mathbf{G}$. Form an $n \times (n-k)$ matrix $\mathbf{H}$ as follows:

1. in the rows of $\mathbf{H}$ corresponding to the leading columns of $\mathbf{G}$, place, in order, the rows of $\mathbf{X}$
2. in the remaining $n-k$ rows of $\mathbf{H}$, place, in order, the rows of the $(n-k) \times (n-k)$ identity matrix $\mathbf{I}_{n-k}$.

Then the columns of $\mathbf{H}$ form a basis for $C^{\perp}$.

Here is a more intuitive description of Algorithm 2.13. Start with matrix $\mathbf{A}$, and use EROS to convert:

$$\mathbf{A} \to \begin{pmatrix} \mathbf{G} \\ \mathbf{0} \end{pmatrix} \text{ in RREF.}$$

Then permute the columns of $\mathbf{G}$ to form $\mathbf{G} \to \mathbf{G}' = (\mathbf{I}_k \ \mathbf{X})$.
Form a matrix $\mathbf{H}'$ as follows:

$$\mathbf{H}' = \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_{n-k} \end{pmatrix}.$$

Apply the inverse of the permutation applied to the columns of $\mathbf{G}$ to the rows of $\mathbf{H}'$ to form $\mathbf{H}$.

Example 2.14 Consider the code $C = \langle S \rangle$ where $S = \{11010, 10001, 01001, 11000\}$. Use Algorithm 2.13 to find a basis for C and a basis for $C^{\perp}$.

$$\mathbf{A} = \begin{pmatrix} 1&1&0&1&0 \\ 1&0&0&0&1 \\ 0&1&0&0&1 \\ 1&1&0&0&0 \end{pmatrix} \to \begin{pmatrix} 1&1&0&1&0 \\ 0&1&0&1&1 \\ 0&1&0&0&1 \\ 0&0&0&1&0 \end{pmatrix} \to \begin{pmatrix} 1&1&0&1&0 \\ 0&1&0&1&1 \\ 0&0&0&1&0 \\ 0&0&0&1&0 \end{pmatrix} \to \begin{pmatrix} 1&1&0&1&0 \\ 0&1&0&1&1 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \end{pmatrix}$$

$$\to \begin{pmatrix} 1&0&0&0&1 \\ 0&1&0&1&1 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \end{pmatrix} \to \begin{pmatrix} \textcircled{1}&0&0&0&1 \\ 0&\textcircled{1}&0&0&1 \\ 0&0&0&\textcircled{1}&0 \\ 0&0&0&0&0 \end{pmatrix} \quad \text{which is in RREF}$$

A basis for C is $\{10001, 01001, 00010\}$.

Now we have $\mathbf{G} = \begin{pmatrix} \textcircled{1}&0&0&0&1 \\ 0&\textcircled{1}&0&0&1 \\ 0&0&0&\textcircled{1}&0 \end{pmatrix}$, so we have $\mathbf{G}' = \left(\begin{array}{ccc|cc} 1&0&0&0&1 \\ 0&1&0&0&1 \\ 0&0&1&0&0 \end{array}\right) = (\mathbf{I}_3 \ \ \mathbf{X})$.

Thus $k = 3$ and $\mathbf{X} = \begin{pmatrix} 0&1 \\ 0&1 \\ 0&0 \end{pmatrix}$.

The rows of $\mathbf{X}$ are placed in the first three rows, respectively, of the $5 \times (5-3)$ matrix $\mathbf{H}'$. The remaining rows of $\mathbf{H}'$ are filled with the the 2×2 identity matrix. Thus

$$\mathbf{H}' = \begin{pmatrix} 0&1 \\ 0&1 \\ 0&0 \\ 1&0 \\ 0&1 \end{pmatrix} \quad \text{and so} \quad \mathbf{H} = \begin{pmatrix} 0&1 \\ 0&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix}$$

Thus a basis for $C^{\perp}$ is $\{00100, 11001\}$.

Exercise 2.15 Verify that for the previous example, $\mathbf{GH} = \mathbf{0}$.

We make a few comments to justify why Algorithm 2.13 works. The $n-k$ columns of $\mathbf{H}$ are linearly independent and $\dim C^\perp = n - \dim C = n-k$, so the columns of $\mathbf{H}$ are a basis for a subspace of the correct dimension. Furthermore,

$$\mathbf{G}'\mathbf{H}' = (\mathbf{I}_k \ \ \mathbf{X}) \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_{n-k} \end{pmatrix} = \mathbf{X} + \mathbf{X} = \mathbf{0}.$$

We apply the same permutation to the columns of $\mathbf{G}'$ and to the rows of $\mathbf{H}'$ to obtain $\mathbf{G}$ and $\mathbf{H}$, so we still get $\mathbf{GH} = \mathbf{0}$. Thus each row of $\mathbf{G}$ is orthogonal to each column of $\mathbf{H}$, and so if the rows of $\mathbf{G}$ form a basis for C then the columns of $\mathbf{H}$ must form a basis for $C^\perp$.

Exercise 2.16 If $S = \{101010, 010101, 111111, 000111, 101100\}$, find a basis B for the code $C = \langle S \rangle$, and find a basis $B^\perp$ for the dual code $C^\perp$. Determine the number of codewords in each of C and $C^\perp$.

$$\begin{pmatrix} 1\,0\,1\,0\,1\,0 \\ 0\,1\,0\,1\,0\,1 \\ 1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1 \\ 1\,0\,1\,1\,0\,0 \end{pmatrix} \longrightarrow \text{row reductions}$$

$$G = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \underline{1}\,0\,1\,0\,1\,0 \\ 0\,\underline{1}\,0\,0\,1\,0 \\ 0\,0\,0\,\underline{1}\,1\,0 \\ 0\,0\,0\,0\,0\,\underline{1} \end{pmatrix} \qquad \begin{matrix} 1 & 2 & 4 & 6 & 3 & 5 \\ \end{matrix} \begin{pmatrix} 1\,0\,0\,0\,1\,1 \\ 0\,1\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0\,0 \end{pmatrix} = (I \mid X)$$

$$H' = \begin{pmatrix} X \\ I \end{pmatrix} = \begin{pmatrix} 1\,1 \\ 0\,1 \\ 0\,1 \\ 0\,0 \\ 1\,0 \\ 0\,1 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 4 \\ 6 \\ 3 \\ 5 \end{matrix} \qquad H = \begin{pmatrix} 1\,1 \\ 0\,1 \\ 1\,0 \\ 0\,1 \\ 0\,1 \\ 0\,0 \end{pmatrix}$$

$|C| = 2^4 = 16$

Basis for $C = \{101010, 010010, 000110, 000001\}$

Basis for $C^\perp = \{101000, 110110\}$ $\quad |C^\perp| = 2^2 = 4$