Topics in Discrete Mathematics
Spring Term 2018/19

ERROR-CORRECTING CODES

ALEXANDER J. MALCOLM

Abstract. These lecture notes form the first topic in the 2018/19 course, Topics in Discrete Mathematics. The second topic, lectured by Dr. Dan Martin, is on cryptography.

Contents

Date: February 14, 2019.

0.1. **Course Summary.**

- **Unit code:** MATH30002 & MATHM0009
- **Level of study:** H/6 & M/7
- **Credit points:** 10
- **Teaching block** (weeks): 2 (13-18)
- **Lecturer:** Alex Malcolm
- **Email:** alex.malcolm@bristol.ac.uk
- **Course homepage:** https://alexmalcolmblog.wordpress.com/teaching/
- **Assessment:** This unit is examined entirely by a single exam in May/June 2019. The pass mark is 40 for MATH30002 and 50 for MATHM0009.
- **Lectures**
  - Tuesdays 10am - SM2, Maths Building
  - Thursdays 5pm - SM1, Maths Building
  - Fridays 5pm - SM1, Maths Building
- **Maths Cafe**
  - Thursdays 11am - Portacabin 2, Maths Building rear (week 14 onwards)

Although these lecture notes are entirely self-contained, outside reading is of course encouraged. There is a strong literature on error-correcting codes, and the following are particularly recommended:

(1) F.J. MacWilliams and N.J.A. Sloane: The Theory of Error-Correcting Codes;
(2) J.H. van Lint: Introduction to Coding Theory;
(3) R.H. Morelos-Zaragoza: The Art of Error-Correcting Coding.

1. Introduction

Transmitting messages is an important practical problem. Information is a valuable asset for everyday life, but when it is shared between people or machines it can pick up errors. For example it is very easy for us to spell a word incorrectly or even use a wrong word entirely. Likewise, messages sent by a computer in binary may pick up an error due to transceiver fault e.g. a "bit-flip" interchanging a 0 for a 1 with a given probability.

Some errors can be *detected* but others cannot. For example, suppose that Alice and Bob are friends, and Bob receives the following message from Alice

Pub tonihgt? The Red Loin at 8pm?

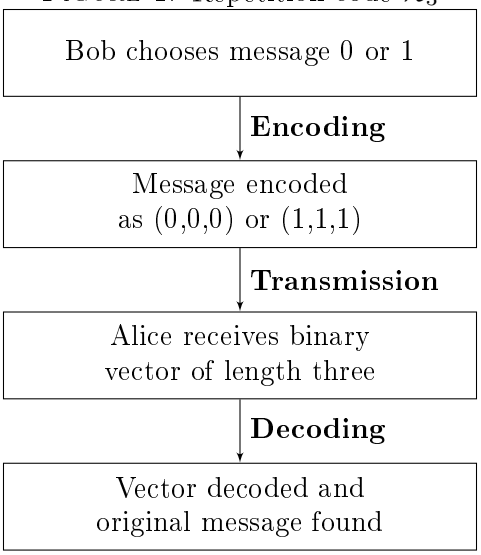
Bob would immediately recognise that an error has been made in the message, but how many? He can easily detect an error in the word "tonihgt" as this isn't in the english language. Furthermore as it very closely resembles the word "tonight", he can be confident that this was the intended message and can *correct* it appropriately. But what about "Red Loin"? This is a strange phrase, but it is correct english. With a little extra information i.e. the knowledge that "Red Lion" is the most common name for an english pub, Bob can probably correct this too. But he can't detect any possible error in the time e.g. changing "8" to "7" would give an equally valid message. To be confident that the time was correct, he would have to assume only two errors have occurred.

To combat problems such as this, we design systems of messages that remain legible even in the presence of errors. These are known as *error-correcting codes*.

A simple example - the Yes/No code: Bob wants to respond to Alice's question, and decides to use a simple system where the message "1" corresponds to answer "yes", and "0" corresponds to "no". But just sending the 1 or 0 is a terrible system: if Bob makes a single mistake and presses the wrong key, the message changes completely. Furthermore, whichever letter Alice receives, she has no way of telling if an error has occurred.

We can improve the process by repeating the choice. Let $\mathcal{C} = \{(0,0,0),(1,1,1)\} \subset \mathbb{F}_2^3 = \{0,1\}^3$, these two vectors will be the *codewords*. The process of transmitting encoded messages is then given by Fig. 1.

FIGURE 1. Repetition code $\mathcal{R}_3$



It should be clear that this system can detect up to two "bit-flip" errors. E.g. if Alice receives the vector $(0,1,0)$ then either one error has occurred (the middle 0 changing to a 1) or two has occurred (the outside 1s turning to 0s). But more importantly, it can also correct one error. How do we do this? Well we take our received vector and count the number of 1s or 0s. If there are more 1s, then the intended message is (1,1,1), and vice versa. This is called *majority logic* decoding; intuitively it makes sense as we would expect a low number of such errors to occur.

Example 1.1. *We can generalise this idea to arbitrary dimensions. Let*

$$\mathcal{R}_n := \{(0,0,\dots 0),(1,1,\dots 1)\} \subset \mathbb{F}_2^n,$$

for any n. This is called the n-bit repetition code. *Intuitively these codes can detect $n-1$ errors and correct $\lfloor (n-1)/2 \rfloor$ errors.*

A drawback of the repetition code is that you can't send many messages.

The following example allows us to send 8 messages while still correcting 1 error.

Example 1.2. ***The simplex code*** *$\mathcal{C}_3$:*

Suppose we want to send the message (x,y,z), a vector of length three where $x,y,z \in \mathbb{F}_2 = \{0,1\}$. Let $\mathcal{C}_3$ be the code consisting of vectors of length 7, $(x,y,z,a,b,c,d) \in \mathbb{F}_2^7$ satisfying the following

$$x+y=a,\ x+z=b,\ y+z=c;$$
$$x+y+z+a+b+c=d.$$

These equations are called the check-sums. So our encoding is $(x,y,z) \to (x,y,z,a,b,c,d)$, and we easily check that there are 8 messages and 8 corresponding codewords.

Suppose we receive (0111011). *Well*

$$x+y=1=a$$
$$x+z=1\neq b$$
$$y+z=0\neq c$$
$$x+y+z+a+b+c\neq d.$$

So there is an error, but where? Well it is in z because this breaks the b, c and d check sums. We correct z, to get corrected codeword (0101011) *corresponding to the message* (010).

Exercise 1.3. *Come up with a decoding/error-correction method to show that $\mathcal{C}_3$ corrects any single error.*

HINT: *Errors in different positions x, y, z, a, b, c, d correspond to different numbers of check-sums failing.*

Now we have seen some examples, we should make the formal definition of a code clear.

Definition 1.4. Let A be a finite set (an "alphabet"). A code of length n over A, is a subset $\mathcal{C} \subseteq A^n$. An element of A^n is called a *vector* (or *word*) and the elements of $\mathcal{C}$ are called *codewords*.

Examples 1.1 and 1.2 are both codes over the alphabet $A = \mathbb{F}_2$. They have lengths n and 7 respectively.

Notation: We will typically write elements of A^n in vector form i.e. $\mathbf{a} = (a_1, \dots, a_n)$ where $a_i \in A$. This is natural as we will often restrict our attention to the case where A is a finite field and A^n is a vector space. However, when $A = \mathbb{F}_2 = \{0, 1\}$ we may also use the shorthand $(0, 1, 1, 1, 0) = 01110$. This shorthand is very common in the literature.

Let's also be clear about the exact nature of an "error".

Definition 1.5. Let $\mathcal{C} \subseteq A^n$ be a code of length n over alphabet A. A single error in a codeword $\mathbf{c} = (c_1, \dots, c_n)$ is a change of one of the components c_j.

Example 1.6. *Suppose that Alice sends the codeword $\mathbf{c} = (1, 0, 1, 1)$. If $\mathbf{c}' = (1, 1, 1, 1)$ is received by Bob, then the codeword has picked up 1 error in the 2nd position. If instead $\mathbf{c}' = (1, 1, 1, 0)$ then there are two errors (in the 2nd and 4th positions).*

Other types of errors are possible e.g. a codeword could be shortened by the nth component disappearing completely. However these will not be the focus of this course.

Roughly speaking, the English language is a code over the alphabet $A = \{a, b, \dots, z\}$ but here the codewords do not have the same length. We could remedy this by padding words with extra zs at the end but this would be very inefficient - the average word length in English is 5, compared to the longest word with 45 letters. Furthermore, comparatively few of the 26^{45} choices in A^{45} are actually real English words. On the other hand, it is this sparsity that allows for relatively easy "decoding" when we make a spelling mistake.

In this course we will focus on the mathematical properties of error-correcting codes (moreso than the encoding/decoding procedures). We'll see that most of the best codes are *linear*, that is, they possess the structure of vector spaces.

2. Linear Algebra Revision

Linear algebra will be an important tool for studying the properties of error-correcting codes. Therefore in this section we shall briefly recap some of the fundamentals of fields and vector spaces. If you need a refresher, there are a few revision exercises on the first problem sheet.

2.1. Fields.

Definition 2.1. A *field* is a set F with two binary operations $+, \times$ such that $(F, +)$ and $(F \backslash \{0\}, \times)$ are abelian groups, and multiplication distributes over addition i.e. $a(b + c) = ab + ac$ for all $a, b, c \in F$.

Intuitively, a field is simply a set on which the operations of *addition, subtraction, multiplication and division* behave "naturally" (i.e. as they do in the real numbers).
There are many examples:

Example 2.2. *The sets $\mathbb{Q}, \mathbb{R}, \mathbb{C}$ are fields. The set of integers modulo a prime number p is also a field, and we denote it by $\mathbb{F}_p$.*

Exercise 2.3. *The sets $\mathbb{Z}, \mathbb{N}$ are not fields. Neither are the integers modulo n, for n composite. Why is this?*

We can build *extension fields* out of old fields.

Definition 2.4. Let X be a set and F a field. Then the *extension* of F by X is is defined to be the smallest field containing F and X. We denote this $F(X)$.

We are only concerned with fields of the form $F(\alpha)$ where α is *algebraic* over the base field F i.e. α is the root of a polynomial with coefficients in F.
For example

Example 2.5.

(1) $\mathbb{Q}(\sqrt{2}) = \{a + b\sqrt{2} \,|\, a, b \in \mathbb{Q}\}$;
(2) $\mathbb{C} = \mathbb{R}(i) = \{a + bi \,|\, a, b \in \mathbb{R}\}$.

Most of the examples listed above are infinite, but in this course we will focus on finite fields. Now it is clear that $\mathbb{F}_p$, the integers mod p, form a finite set but these are not the only examples.
Fact:

(1) Every finite field is of the form $\mathbb{F}_p(\alpha)$ where p is a prime, and α is the root of an irreducible polynomial f, over $\mathbb{F}_p$. If degree$(f) = l$, then

$$\mathbb{F}_p(\alpha) = \{a_{l-1}\alpha^{l-1} + a_{n-2}\alpha^{l-2} + \cdots + a_1\alpha + a_0 \,|\, a_i \in \mathbb{F}_p\}. \tag{1}$$

(2) It's easy to see that $|\mathbb{F}_p(\alpha)| = p^l$. But in fact, there is exactly one finite field of size p^l up to isomorphism, for each p, and l. We denote this field by $\mathbb{F}_{p^l}$ - usually we define $q = p^l$ and use the shorthand $\mathbb{F}_q$.

Example 2.6. *Let's construct the finite field of 4 elements i.e. $\mathbb{F}_4 = \mathbb{F}_{2^2}$: Well $f = x^2 + x + 1 \in \mathbb{F}_2[x]$ is irreducible (check $f(0), f(1) \neq 0$) and degree$(f) = 2$ as required. Let α be a root of f. Then*

$$\mathbb{F}_4 = \{a_1\alpha + a_0 \,|\, a_i \in \mathbb{F}_2\} = \{0, 1, \alpha, \alpha + 1\}.$$

To see that this field is closed under multiplication, simply note that powers of α can be reduced by applying the rule $\alpha^2 = \alpha + 1$ (as given by f).

Remark. It's important to note from (1) that all the coefficients a_i are contained in $\mathbb{F}_p$. Therefore any addition or multiplication carried out in the finite field is done "modulo p".The prime p is known as the *characteristic* of the field.
Warning! The finite field $\mathbb{F}_{p^l}$ is **not** the same as integers mod p^l!

2.2. **Vector Spaces.** Just how fields were defined to be analogous to $\mathbb{Q}$ or $\mathbb{R}$, we want an abstract notion of *vector space* that captures the essential properties of n-dimensional real space $\mathbb{R}^n$.

Definition 2.7. A *vector space* over a field F is a non-empty subset V with two operations; *addition* and *scalar multiplication* by elements of F such that:

(1) (V,+) is an abelian group;

(2) $\alpha(\beta\mathbf{v}) = (\alpha\beta)\mathbf{v}$ for all $\alpha, \beta \in F$ and $\mathbf{v} \in V$;
(3) $1\mathbf{v} = \mathbf{v}$ for all $\mathbf{v} \in V$;
(4) $\alpha(\mathbf{v} + \mathbf{v}') = \alpha\mathbf{v} + \alpha\mathbf{v}'$ for all $\alpha \in F$ and $\mathbf{v}, \mathbf{v}' \in V$;
(5) $(\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$ for all $\alpha, \beta \in F$ and $\mathbf{v} \in V$.

Again, there are many natural (and hopefully familiar) examples:

Example 2.8.

(1) Vectors: F^n is a vector space over F under the usual addition and scaling of vectors.
(2) Matrices: $M_{a,b}(F)$ is a vector space over F under the usual addition and scaling operations of matrices.
(3) Polynomials: $F[x]$ is a vector space over F under the usual addition and scaling operations of polynomials.

We have the natural inclusion of vector spaces through the notion of *subspace*:

Definition 2.9. Let V be a vector space over field F, and let W be a subset of V. We say that W is a *subspace* of V if W is closed under addition, and scalar multiplication by F. I.e. if $\alpha\mathbf{w} + \beta\mathbf{w}' \in W$ for all $\alpha, \beta \in F$ and $\mathbf{w}, \mathbf{w}' \in W$.

Notation: In some texts the symbol "$\subseteq$" is used to denote inclusion of subspaces, but this can be ambiguous and so we **do not** do that here. In this course "$\subseteq$" will only indicate inclusion as sets. We shall use "$\leq$" to denote inclusion of vector spaces.

Example 2.10.

(1) Let $V = \mathbb{R}^2$. Then $W = \{a(1,0) \,|\, a \in \mathbb{R}\} \leq V$.
(2) Let $V = F[x]$ for some field F, and let W denote the polynomials in V of degree at most 3. Then $W \leq V$.
(3) Let V be a vector space and let $X := \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subseteq V$ be a collection of vectors. Then the span of X over F, defined

$$\langle X \rangle_F := \langle \mathbf{v}_1, \dots, \mathbf{v}_k \rangle_F := \{\sum_{i=1}^{k} a_i \mathbf{v}_i \,|\, a_i \in F\},$$

is a vector subspace of V.

When studying $\mathbb{R}^n$, we usually fix a *co-ordinate system* to allow for an easy description of vectors. For example it is common to use *cartesian co-ordinates* $\mathbf{e}_i := (0, \dots, 0, 1, 0, \dots, 0)$ so that $\mathbb{R}^n = \langle \mathbf{e}_1, \dots, \mathbf{e}_n \rangle_{\mathbb{R}}$. Furthermore, every vector $\mathbf{v}$ has a unique decomposition $\mathbf{v} = a_1\mathbf{e}_1 + \dots a_n\mathbf{e}_n$.
Now there are many other choices of spanning vectors for $\mathbb{R}^n$ but note that the set of $\mathbf{e}_i$ has the stronger property that there is no *redundancy*: if we remove $\mathbf{e}_1$ (or indeed any $\mathbf{e}_i$) then it's clear that $\mathbb{R}^n \neq \langle \mathbf{e}_2, \dots, \mathbf{e}_n \rangle_{\mathbb{R}}$.

Example 2.11. *Let $V = \mathbb{R}^2$ and consider $X_1 = \{(1,0), (0,1)\}$, $X_2 = \{(1,0), (0,1), (1,1)\}$. Clearly $\langle X_1 \rangle_{\mathbb{R}} = \langle X_2 \rangle_{\mathbb{R}}$, but decompositions in the second spanning set X_2 are not unique! For a general $\mathbf{v} \in \mathbb{R}^2$ there will be multiple solutions $a, b, c \in \mathbb{R}$ to*

$$\mathbf{v} = a(1,0) + b(0,1) + c(1,1).$$

*This is because we have a redundancy or **dependence** in X_2: clearly $(1,1) = (1,0) + (0,1)$ and so we don't really need this extra vector.*

This motivates the following definition

Definition 2.12. Let V be a vector space and $\mathbf{v}_1, \dots, \mathbf{v}_n \in V$.

- We say that $\mathbf{v}_1, \ldots, \mathbf{v}_n \in V$ are *linearly independent* if the only solution to the equation $a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n = \mathbf{0}$ is $a_i = 0$ for all i.
- We say that $\mathbf{v}_1, \ldots, \mathbf{v}_n \in V$ is a *basis* for V if they are linearly independent and $\langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle_F = V$.

Example 2.13. *Let F be a field and let $V = F^n$ be the vector space of n-dimensional vectors over F. Then $\mathbf{e}_1, \ldots, \mathbf{e}_n$ (where $\mathbf{e}_i := (0, \ldots, 0, 1, 0, \ldots, 0)$ as before) form a basis of V. We call $\mathbf{e}_i$ the **standard basis vectors** of F^n.*

Fact: If a vector space has a finite basis then all bases have the same size - this is known as the *dimension* of the vector space.

This is exactly why we had the redundancy in the spanning set X_2, in Example 2.11 above. As $\dim(\mathbb{R}^2) = 2$, any set of 3 or more (non-zero) vectors will not be linearly independent.

We can use matrices to generate vector spaces (over a field F) in the following two ways. For $M \in M_{a,b}(F)$:

- Let $\mathbf{r}_i$, $1 \leq i \leq a$, denote the rows of M. Then $\text{RowSpace}(M) = \langle \mathbf{r}_1, \ldots, \mathbf{r}_a \rangle_F \leq F^b$. The dimension of this space is exactly $\text{rank}(M)$. **Warning!** $\text{rank}(M) \leq a$; it is not necessarily equal to a i.e. the number of rows in M
- Recall that $\text{NullSpace}(M)$ is the set of vectors such that $M\mathbf{v}^T = 0$. Furthermore $\text{NullSpace}(M) \leq F^b$ and has dimension, $\text{nullity}(M)$ called the *nullity* of M.

Lastly, let's recall an important result from linear algebra that we'll use (but not prove) repeatedly in this course.

Theorem 2.14. ***The Rank-Nullity Theorem*** *Let $M \in M_{a,b}(F)$. Then $\text{rank}(M) + \text{nullity}(M) = b$.*

Intuitively, what the rank-nullity theorem is saying is that if you have a set of k independent homogeneous linear equations in n variables, then you expect the space of solutions to be $(n-k)$-dimensional.

3. Linear codes

In Section 1 we defined a code and saw some examples. Now, after the revision of Section 2, we are ready to study these codes in further detail.

3.1. Preliminaries.

Definition 3.1. Let $\mathbf{u}, \mathbf{v} \in A^n$. We define the Hamming distance (usually shortened simply to "distance") between the two vectors to be the number of co-ordinates in which they differ i.e.

$$d(\mathbf{u}, \mathbf{v}) := |\{i \,|\, u_i \neq v_i\}|.$$

Example 3.2. *Some distances in $\mathbb{F}_2^5$:*

$$d(01110, 01001) = 3;$$
$$d(11111, 11001) = 2.$$

The Hamming distance on A^n satisfies a number of natural properties.

Lemma 3.3. *The Hamming distance on A^n is a **metric**. That is, it satisfies the following three properties:*

(1) $d(\mathbf{u}, \mathbf{v}) \geq 0$, with equality if and only if $\mathbf{u} = \mathbf{v}$;
(2) $d(\mathbf{u}, \mathbf{v}) = d(\mathbf{v}, \mathbf{u})$;
(3) $d(\mathbf{u}, \mathbf{v}) \leq d(\mathbf{u}, \mathbf{s}) + d(\mathbf{s}, \mathbf{v})$ for all $\mathbf{s} \in A^n$.

Proof. Left as an exercise. □

It is of course also natural to consider the *minimum* possible distance between any two codewords in $\mathcal{C}$, as well as *spheres* centred at vectors:

Definition 3.4. Let $\mathcal{C} \subseteq A^n$ be a code. The minimum distance $d(\mathcal{C})$ of $\mathcal{C}$ is

$$d(\mathcal{C}) := \min\{d(\mathbf{c}, \mathbf{d}) \,|\, \mathbf{c}, \mathbf{d} \in \mathcal{C},\, \mathbf{c} \neq \mathbf{d}\}.$$

Definition 3.5. Let $\mathbf{v} \in A^n$. The *Hamming sphere* with centre $\mathbf{v}$ and radius $r \geq 0$ is the collection of vectors

$$S_r(\mathbf{v}) := \{\mathbf{u} \in A^n \,:\, d(\mathbf{u}, \mathbf{v}) \leq r\}.$$

As you probably expect, the notion of distance is intrinsically linked to the error-detection/correction properties of a given code. Suppose Alice wants to message Bob: she sends the codeword $\mathbf{c}$ but some errors occur and Bob receives a vector $\mathbf{v}$. Intuitively, if $\mathbf{v}$ is "close" to multiple codewords it is going to be hard to decide which is the correct codeword that Alice sent.

Definition 3.6. Let $\mathcal{C} \subseteq A^n$ be a code.
We say that $\mathcal{C}$ *detects* r errors if for each $\mathbf{c} \in \mathcal{C}$, changing any r letters does not produce another codeword. I.e.

$$S_r(\mathbf{c}) \cap \mathcal{C} = \{\mathbf{c}\}, \text{ for all } \mathbf{c} \in \mathcal{C}.$$

We say that $\mathcal{C}$ *corrects* r errors if for each $\mathbf{v} \in A^n$, changing up to r letters produces at most one codeword $\mathbf{c} \in \mathcal{C}$. I.e.

$$|S_r(\mathbf{v}) \cap \mathcal{C}| \leq 1, \text{ for all } \mathbf{v} \in A^n.$$

Example 3.7. *Recall the binary repetition code $\mathcal{R}_n$ from Example 1.1. We see that this indeed detects $n-1$ errors and corrects $\lfloor (n-1)/2 \rfloor$ errors, by the above definitions.*

Notice from this example, the importance of "at most one codeword" in the definition of correction. For example, if $\mathcal{C} = \mathcal{R}_6$ then we can clearly correct up to 2 errors but $|S_2(000111) \cap \mathcal{C}| = 0$.

The first theorem of this course confirms the intuition described above, and highlights the fundamental importance of minimum distance.

Theorem 3.8. *Let $\mathcal{C} \subseteq A^n$ be a code with minimum distance $d(\mathcal{C}) = d$.*

(1) $\mathcal{C}$ detects r errors if and only if $r \leq d-1$.
(2) $\mathcal{C}$ corrects r errors if and only if $r \leq \frac{d-1}{2}$.

Proof.

(1) Here the proof is a simple unpacking of the two definitions: $\mathcal{C}$ detects r-errors if and only if $S_r(\mathbf{c}) \cap \mathcal{C} = \{\mathbf{c}\}$, for all $\mathbf{c} \in \mathcal{C}$. This is equivalent to $d(\mathbf{c}, \mathbf{c}') > r$ for all $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$. And this is true if and only if $r < d(\mathcal{C}) = d$.
(2) Firstly suppose that $d(\mathcal{C}) \geq 2r+1$ and assume for a contradiction that $\mathcal{C}$ does not correct r errors. Then by definition there exists $\mathbf{v} \in A^n$ such that $|S_r(\mathbf{v}) \cap \mathcal{C}| \geq 2$. Take two distinct codewords $\mathbf{c}, \mathbf{c}' \in S_r(\mathbf{v}) \cap \mathcal{C}$. It follows by the triangle inequality that
$$d \leq d(\mathbf{c}, \mathbf{c}') \leq d(\mathbf{c}, \mathbf{v}) + d(\mathbf{v}, \mathbf{c}') \leq 2r \leq d-1.$$
This gives the desired contradiction, proving the right to left implication of (2). The reverse implication is left as an exercise.

□

Definition 3.9. From Theorem 3.8, we see that a linear code with minimum distance $d(\mathcal{C}) = d$ can correct at most $e := \lfloor \frac{d-1}{2} \rfloor$ errors. We call e the *error correcting index* of $\mathcal{C}$.

Looking at Theorem 3.8 we might think that the best thing to do is pick codes that maximise distance, as this will yield the ability to correct/detect large errors. However as the repetition code $\mathcal{R}_n$ demonstrates, this isn't always the best idea - it has the maximal possible distance for a code of length n but we can only send two messages! So other factors can be important; we'll discuss this in further detail in Section 6.

However to make it easier to study their properties, we can start by restricting our attention to codes that have at least some simple mathematical structure. It turns out that most of the best codes are vector subspaces of $\mathbb{F}_q^n$ and that introducing this "modest" condition leads to significant developments in the theory.

Definition 3.10. We call $\mathcal{C} \leq \mathbb{F}_q^n$ a *linear code* if $\mathcal{C}$ is a vector subspace of $\mathbb{F}_q^n$. If $\mathcal{C}$ is a subspace of dimension k, then we correspondingly say that $\mathcal{C}$ has dimension k, or is an $[n,k]_q$-linear code. Furthermore, if it also has minimum distance $d = d(\mathcal{C})$, we call it an $[n,k,d]_q$-linear code.

The $[n,k,d]_q$-notation is simply a concise way of denoting four fundamental properties of the code.

Remark. **From this point on, all codes considered in this course will be linear codes over finite fields. That is, alphabets $A = \mathbb{F}_q$ for a prime power q, and $\mathcal{C} \leq \mathbb{F}_q^n$.**

Now what can we immediately deduce about linear codes? Firstly, recall from Definition 1.4 that a general code can be of any size between 1 and $|A|^n$. But for linear codes there are far fewer possible sizes.

Lemma 3.11. *Let $\mathcal{C}$ be an $[n,k]_q$- linear code. Then $|\mathcal{C}| = q^k$.*

Proof. We simply need to show that a subspace $\mathcal{C} \leq \mathbb{F}_q^n$ of dimension k, has size q^k. Well as it is a vector space, $\mathcal{C}$ has a basis, say $\mathbf{c}_1, \ldots, \mathbf{c}_k$. It follows that

$$\mathcal{C} = \{\lambda_1\mathbf{c}_1 + \cdots + \lambda_k\mathbf{c}_k \,|\, \lambda_i \in \mathbb{F}_q\}.$$

As the basis vectors $\{\mathbf{c}_i\}$ are linearly independent, any choice of scalars $(\lambda_1, \ldots, \lambda_k)$ produces a distinct vector in $\mathcal{C}$. But there are q^k such choices and therefore $|\mathcal{C}| = q^k$. □

Another immediate advantage of linear codes is that it is much quicker to find the minimum distance. For a general code, we need to compare the distance between every two distinct codewords i.e. compute $\binom{|\mathcal{C}|}{2}$ distances. We claim that if $\mathcal{C}$ is linear, then the work can be reduced to $|\mathcal{C}|-1$ calculations; far more efficient. To see this we use the notion of codeword *weight*.

Definition 3.12. The *weight* of a vector $(v_1, \ldots, v_n) = \mathbf{v} \in \mathbb{F}_q^n$, denoted $\mathrm{wt}(\mathbf{v})$, is

$$\mathrm{wt}(\mathbf{v}) := |\{i \,|\, v_i \neq 0\}| = d(\mathbf{v}, 0).$$

Lemma 3.13. *Let $\mathcal{C}$ be an $[n,k]_q$-linear code. Then*

$$d(\mathcal{C}) = \min\{wt(\mathbf{c}) \,|\, \mathbf{c} \neq \mathbf{0}\}.$$

Proof. Firstly observe that as $\mathcal{C}$ is linear, if $\mathbf{c}, \mathbf{c}' \in \mathcal{C}$ then $\mathbf{c} - \mathbf{c}' \in \mathcal{C}$. Furthermore, $\mathbf{c} - \mathbf{c}'$ has non-zero entries precisely in the positions where $\mathbf{c}$ and $\mathbf{c}'$ differ. Hence $d(\mathbf{c}, \mathbf{c}') = d(\mathbf{c} - \mathbf{c}', 0) = \mathrm{wt}(\mathbf{c} - \mathbf{c}')$. So the distance between any two codewords can be realised as the weight of another codeword. The result follows. □

Remark. When calculating $d(\mathcal{C})$ with this method, it is important to range over **all** codewords in $\mathcal{C}$. Taking a minimum over a basis is not enough!

E.g. consider the $[5,2]_2$-linear code $\mathcal{C}$ spanned by $\mathbf{c}_1 = (1,1,1,1,1)$ and $\mathbf{c}_2 = (1,1,0,0,1)$. Clearly $\mathrm{wt}(\mathbf{c}_1) = 5$ and $\mathrm{wt}(\mathbf{c}_2) = 3$, but $\mathbf{c}_1 - \mathbf{c}_2 \in \mathcal{C}$ and $\mathrm{wt}(\mathbf{c}_1 - \mathbf{c}_2) = 2$.

3.2. **Two presentations of a linear code.** Linear codes are convenient in that they can be easily described using a basis (this is far more efficient than writing down all the codewords!). We store the basis in a *generator matrix*:

Definition 3.14. Let $\mathcal{C}$ be an $[n,k]_q$-code. Then a *generator matrix* for $\mathcal{C}$ is a matrix $G \in M_{k,n}(\mathbb{F}_q)$ whose rows form a basis for $\mathcal{C}$. I.e. $\mathcal{C} = \text{RowSpace}(G)$ and $\text{Rank}(G) = \dim(\mathcal{C}) = k$.

So we can describe all codewords in $\mathcal{C}$ by taking linear combinations of the basis vectors; this is achieved through easy matrix multiplication:

Lemma 3.15. *Let $\mathcal{C}$ be an $[n,k]_q$-code with generator matrix G. Then*

$$\mathcal{C} = \{\mathbf{w}G \,|\, \mathbf{w} \in \mathbb{F}_q^k\}.$$

The matrix multiplication $\mathbf{w}G$ can be viewed as the encoding process: we start with a message $\mathbf{w}$ which has length k as it lies in the *message space* $\mathbb{F}_q^k$. Then multiplying by G encodes the message to give $\mathbf{w}G = \mathbf{c}$, a codeword that lives in the codespace $\mathcal{C}$. The ratio k/n can then be viewed as the proportion of useful (non-redundant) information contained in a codeword. The value k/n is called the *rate* of $\mathcal{C}$.

Exercise 3.16. *Let $\mathcal{C}$ be an $[n,k]_q$-code with generator matrix G. Show that if $\mathbf{w}G = \mathbf{w}'G$ then $\mathbf{w} = \mathbf{w}'$. Why is this important for the encoding process?*

Example 3.17. *Some examples*

(1) *$\mathcal{R}_5$ - the repitition code of length 5, has generator matrix*

$$G = (1\,1\,1\,1\,1).$$

Here it is easy to see how the encoding process works: we let $\mathbf{w} = 0$ or 1, and we get output code words (00000) *and* (11111) *respectively.*

(2) *Recall the simplex code $\mathcal{C}_3$, as defined in Example 1.2. A natural choice of generator matrix is*

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

But there are other basis choices for this subspace and therefore other choices of generator matrix e.g.

$$G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

would also work.

So evidently, there is not always a unique generator matrix. In fact we have the following:

Exercise 3.18. *Let $\mathcal{C}$ be an $[n,k]_q$-code and suppose that G is a unique generator matrix for $\mathcal{C}$. Show that $n = k = 1$ and $q = 2$.*

More generally we can ask

Exercise 3.19. *How many generator matrices does an $[n,k]_q$-code have?*

If we perform Gaussian elimination on a given generator matrix, we get the so-called *standard form*:

Definition 3.20. We say that a generator matrix is in *standard form* if it is of the form $G = (I_k \, A)$, where I_k is the $k \times k$ identity matrix and $A \in M_{k,n-k}(\mathbb{F}_q)$.

The generator matrix is the first convenient way of defining a linear code. There is however a second method, where rather than specifying a basis, we specify a set of equations that the co-ordinates of the codewords must solve. We describe these equations in the form of a *parity check matrix*, usually denoted H. Now, a codeword solving our equations is equivalent to being in the *null space* of H.

Definition 3.21. We say that a matrix $H \in M_{n-k,n}(\mathbb{F}_q)$ is a *parity check matrix* for the $[n,k]_q$-code $\mathcal{C}$ if

$$\mathcal{C} = \{\mathbf{v} \in \mathbb{F}_q^n \mid H\mathbf{v}^T = 0\},$$

i.e. $\mathcal{C}$ =NullSpace(H).

Example 3.22. *Consider the binary 4-bit repetition code $\mathcal{R}_4$. A vector $\mathbf{v} \in \mathbb{F}_2^4$ is a codeword if and only if $v_i = v_j$ for all $1 \leq i, j \leq 4$. Throwing out unnecessary equations (recall that $\dim(\mathcal{R}_4) = 1$ and hence by rank-nullity we require 3 independent equations) we have that $\mathbf{v} \in \mathcal{R}_4$ if and only if*

$$v_1 = v_2,\ v_2 = v_3 \text{ and } v_3 = v_4.$$

This then gives the parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

We can clearly generalise this to n dimensions. Let $H \in M_{n-1,n}(\mathbb{F}_2)$ be the matrix with entries $H_{i,i} = H_{i,i+1} = 1$, $1 \leq i \leq n-1$ and zero otherwise. Then H is a parity check matrix for $\mathcal{R}_n$.

A great property of the parity check matrix formulation of a code, is that it is very quick to test if a vector is a codeword or not. We simply compute $H\mathbf{v}^T$ and check that it's the zero vector. The trade-off is that using H is not as efficient at producing many codewords as the generator matrix G.

It is easy to see that given any matrix $H \in M_{n-k,n}(\mathbb{F}_q)$ of full rank, we can construct an $[n,k]_q$-code. We simply define $\mathcal{C} := \{\mathbf{v} \in \mathbb{F}_q^n \mid H\mathbf{v}^T = 0\}$, and as the null space of a matrix is closed under addition/scalar multiplication, $\mathcal{C}$ is a subspace of $\mathbb{F}_q^n$ and by definition a linear code. It again has the required rank by the rank-nullity theorem (Thm. 2.14).

Example 3.23. *Consider the linear code $\mathcal{C} \leq \mathbb{F}_5^4$ defined by parity check matrix*

$$H = \begin{pmatrix} 1 & 2 & 3 & 0 \\ 0 & 0 & 4 & 1 \end{pmatrix}.$$

So $\mathcal{C}$ consists of vectors $(v_1, \ldots, v_4) \in \mathbb{F}_5^4$ such that

$$v_1 + 2v_2 + 3v_3 = 0 = 4v_3 + v_4.$$

We first see that the latter of these two conditions is equivalent to $v_3 = v_4$ (we are working mod 5). Now, letting $(v_2, v_3) \in \{(1,0),(0,1)\}$ we get from the first condition that $c_1 = (3,1,0,0)$ and $c_2 = (2,0,1,1)$ are both in $\mathcal{C}$. Thus, as c_1 and c_2 are clearly linearly independent, they form a basis of $\mathcal{C}$, and they can therefore can be used to define a generator matrix for the code

$$G = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 \end{pmatrix}.$$

Exercise 3.24. *Try some more examples of your own. What happens if $H \in M_{n-k,n}(\mathbb{F}_q)$ but is not of full rank?*

4. Hamming codes and some decoding

We can now define one of the most well-known families of error-correcting codes, the Hamming codes.

Definition 4.1. Let q be a prime power, $k \in \mathbb{N}_{\geq 1}$ and $n = n(q,k) := (q^k - 1)/(q-1)$. A Hamming code, $\mathcal{H}_k(q)$ is the $[n, n-k]_q$-linear code for which the parity check matrix has columns that form a maximal set of pairwise linearly independent vectors in $\mathbb{F}_q^k$.

Exercise 4.2. *Double-check that a code defined by such a parity check matrix does indeed have length $n(q,k)$.*

For example consider a binary Hamming code of rank 3 i.e. $\mathcal{H}_3(2)$. This is the $[7,4]_2$-linear code $\mathcal{H}_3(2) = \{\mathbf{v} \in \mathbb{F}_2^7 \,|\, H\mathbf{v}^T = 0\}$, where H has columns consisting of all non-zero vectors of $\mathbb{F}_2^3$ (recall that over $\mathbb{F}_2$, two vectors are linearly independent if and only if they are non-equal). So for example

$$H := \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

The binary Hamming codes are the most commonly found Hamming codes in the literature (and also in this course). We shall therefore often drop the field value $q = 2$ in the notation i.e. we'll typically write $\mathcal{H}_k(2) = \mathcal{H}_k$.

Now we could have ordered the columns in H differently. This would produce a different but *equivalent* (we'll define this later in the course, see Definition 7.2) Hamming code.

We can exhibit a method of correcting 1 error using the Hamming codes.

Firstly, recall that for an $[n,k]_q$- linear code $\mathcal{C}$ with parity check matrix H, $\mathbf{c} \in \mathcal{C}$ if and only if $H\mathbf{c}^T = 0$. In other words, if $\mathbf{v} \in \mathbb{F}_q^n$ such that $H\mathbf{v}^T \neq 0$ then we know that $\mathbf{v} \notin \mathcal{C}$.

Definition 4.3. Let $\mathcal{C}$ be an $[n,k]_q$- linear code with parity check matrix H, and let $\mathbf{v} \in \mathbb{F}_q^n$. We call $H\mathbf{v}^T$ *the syndrome* of $\mathbf{v}$.

The value of the syndrome is important: firstly, to restate the above, a given vector is a codeword if and only if it has zero syndrome. Furthermore, suppose that Alice and Bob are communicating using a linear code $\mathcal{C}$; Alice sends a codeword $\mathbf{c}$ but it picks up an error(s) and Bob receives vector $\mathbf{v} \neq \mathbf{c}$. Provided not too many errors have occurred, we can use the syndrome of $\mathbf{v}$ to work out the original codeword $\mathbf{c}$.

Let's demonstrate how to do this with one error:

Proposition 4.4. *Let $\mathcal{C} \leq \mathbb{F}_q^n$ be a linear code with parity check matrix H.*

1. *Suppose that every set of $m-1$ columns of H is linearly independent. Then the minimum distance $d(\mathcal{C}) \geq m$.*
2. *Suppose in addition to (1) that H contains a set of m linearly dependent columns. Then $d(\mathcal{C}) = m$.*

Proof.

(1) Suppose for a contradiction that $d(\mathcal{C}) \leq m-1$. So by Lemma 3.13 there exists a codeword $0 \neq \mathbf{c} \in \mathcal{C}$ such that $\mathrm{wt}(\mathbf{c}) = r \leq m-1$. Now we can write $\mathbf{c}$ as a sum of r standard basis vectors

$$\mathbf{c} = \lambda_{i_1}\mathbf{e}_{i_1} + \cdots + \lambda_{i_r}\mathbf{e}_{i_r},$$

for some $\lambda_{i_j} \in \mathbb{F}_q^*$. Hence

$$\begin{aligned} 0 = H\mathbf{c}^T &= H(\lambda_{i_1}\mathbf{e}_{i_1}^T) + \cdots + H(\lambda_{i_r}\mathbf{e}_{i_r}^T) \\ &= \sum_{j=1}^{r} \lambda_{i_j} \times i_j\text{-th column of } H. \end{aligned}$$

This is clearly a contradiction against the hypothesis of (1).

(2) Suppose that columns $i_1, \ldots, i_m$ are linearly dependent. Then there exists $\lambda_{i_j} \in \mathbb{F}_q$ such that

$$\sum_{j=1}^{m} \lambda_{i_j} \times i_j\text{-th column of } H = 0.$$

As any $m-1$ subset of these columns is linearly independent by (1), all of $\lambda_{i_j} \neq 0$. Hence

$$\begin{aligned} 0 &= \sum_{j=1}^{m} \lambda_{i_j} \times i_j\text{-th column of } H \\ &= \sum_{j=1}^{m} H(\lambda_{i_j}\mathbf{e}_{i_j}^T) = H(\sum_{j=1}^{m} \lambda_{i_j}\mathbf{e}_{i_j}^T). \end{aligned}$$

Therefore $\mathbf{c} = \sum_{j=1}^{m} \lambda_{i_j}\mathbf{e}_{i_j} \in \mathcal{C}$ and $\mathrm{wt}(\mathbf{c}) = m$.

□

Corollary 4.5. *The Hamming codes correct exactly one error.*

Proof. Let H be a parity check matrix for $\mathcal{H}_k(q)$. By Theorem 3.8 and Proposition 4.4, it suffices to show that every pair of columns in H is linearly independent, but that there exist three linearly dependent columns. Well, the former condition follows immediately from the definition of the Hamming codes. It is also clear by definition that amongst its columns, H contains scalar multiples of the vectors $\mathbf{e}_1, \mathbf{e}_2$ and $\mathbf{e}_1 + \mathbf{e}_2$. These columns satisfy the latter condition and we are done.

□

So the Hamming codes correct one error in theory, but how do we actually complete the decoding in practice? Well suppose that a codeword $\mathbf{c} \in \mathcal{H}_k(q)$ is sent but it picks up one error during transmission. That is we receive the vector $\mathbf{v} = \mathbf{c} + \lambda\mathbf{e}_i$ where $\lambda \in \mathbb{F}_q$ and $\mathbf{e}_i$ is the i-th standard basis vector. We compute the syndrome of $\mathbf{v}$:

$$\begin{aligned} H\mathbf{v}^T &= H(\mathbf{c} + \lambda\mathbf{e}_i)^T = H\mathbf{c}^T + \lambda H\mathbf{e}_i^T \\ &= \lambda H\mathbf{e}_i^T \\ &= \lambda \times i\text{-th column of } H. \end{aligned}$$

But the columns of H are pair-wise independent and so it is easy to read off both i and λ (i.e. identify $\mathbf{e}$) by comparing with H. We then finally make the correction $\mathbf{v} - \mathbf{e}$ to retrieve the original codeword.

Correcting more than one error

The method for the Hamming codes is indicative of how to correct more than one error. Firstly, note that similarly to how all codewords $\mathbf{c} \in \mathcal{C}$ have zero syndrome, the syndromes of vectors outside of the code-space are also non-unique.

Lemma 4.6. *Two vectors* $\mathbf{u}, \mathbf{v}$ *have the same syndrome if and only if* $\mathbf{u} - \mathbf{v} \in \mathcal{C}$.

Proof. An easy exercise. □

It follows that the syndrome of a vector $\mathbf{v}$ is uniquely determined by its *coset*

$$\mathbf{v} + \mathcal{C} = \{\mathbf{v} + \mathbf{c} | \mathbf{c} \in \mathcal{C}\}.$$

We can however distinguish some vectors by their syndromes, using the following result.

Proposition 4.7. *Let $\mathcal{C}$ be a linear code with error correcting index e. Then the syndromes of vectors of weight at most e are distinct.*

Proof. Assume for a contradiction that the statement is false; that is, there exist $\mathbf{u}, \mathbf{v}$ with the same syndrome such that $\mathrm{wt}(\mathbf{u}), \mathrm{wt}(\mathbf{v}) \leq e$. It follows that $\mathbf{u} - \mathbf{v} = \mathbf{c}$ for some $\mathbf{c} \in \mathcal{C}$ by Lemma 4.6. But then

$$\begin{aligned} d \leq \mathrm{wt}(\mathbf{c}) = \mathrm{wt}(\mathbf{u} - \mathbf{v}) &= d(\mathbf{u}, \mathbf{v}) \\ &\leq d(\mathbf{u}, \mathbf{0}) + d(\mathbf{0}, \mathbf{v}) \\ &= \mathrm{wt}(\mathbf{u}) + \mathrm{wt}(\mathbf{v}) \\ &\leq 2e \\ &\leq d - 1. \end{aligned}$$

□

This result allows us to design an algorithm for correcting e errors or less.

(1) Find a set of minimal weight representatives $\mathbf{v}_i$ for the cosets of $\mathcal{C}$. These are called the *coset leaders*.
(2) Compute the syndromes of the coset leaders $\mathbf{v}_i$ and store these in a look-up table whose rows are indexed by syndrome.
(3) The error-correction process is as follows: Suppose we receive a message $\mathbf{v}$ with at most e errors. Compute the syndrome $H\mathbf{v}^T$ - this corresponds uniquely to a row of our table i.e. a unique coset leader $\mathbf{v}_j$.
(4) It follows that $\mathbf{v} = \mathbf{v}_j + \mathbf{c}$ for a unique codeword $\mathbf{c} \in \mathcal{C}$. This $\mathbf{c}$ is the intended message.

5. Dual codes

We have seen that any given matrix can be treated as a parity check matrix, defining a linear code. But is the reverse always possible? I.e. does a linear code always have a parity check matrix that defines it? The answer is yes, but to see this we first need to develop the notion of duality. Furthermore we shall see that duality provides a link between the two methods of constructing a code, in particular how the roles of generator matrix G, and parity check H are reversed by "taking duals".

Definition 5.1. Let $\mathcal{C} \leq \mathbb{F}_q^n$ be a linear code. Define the *dual code* to $\mathcal{C}$, denoted $\mathcal{C}^\perp$, by

$$\mathcal{C}^\perp := \{\mathbf{v} \in \mathbb{F}_q^n \,|\, \mathbf{c}\mathbf{v}^T = 0, \forall \mathbf{c} \in \mathcal{C}\}.$$

Exercise 5.2. *Prove that $\mathcal{C}^\perp$ is also a linear code.*

Example 5.3. *Consider the linear code*

$$\mathcal{C} = \{(0,0,0), (1,1,0), (0,1,1), (1,0,1)\} \leq \mathbb{F}_2^3.$$

Suppose that $\mathbf{v} = (v_1, v_2, v_3) \in \mathcal{C}^\perp$. By computing $\mathbf{c}\mathbf{v}^T$ for the 2nd and 3rd codewords in $\mathcal{C}$, it follows that $v_1 = v_2 = v_3$. It is then easy to check that $\mathcal{C}^\perp = \{(0,0,0), (1,1,1)\}$.

Remark. We should be careful not to think of $\mathcal{C}^\perp$ as an orthogonal complement in the sense of real vector spaces - it is quite possible that over a finite field $\mathcal{C} \cap \mathcal{C}^\perp$ is more than just $\{\mathbf{0}\}$. In fact, it is possible that $\mathcal{C} = \mathcal{C}^\perp$, in which case $\mathcal{C}$ is called a *self-dual* code.

Exercise 5.4. *See if you can construct some self-dual codes of your own. What elementary properties of such codes can be deduced?*

Rather than calculating the dot product with all codewords, we can in fact check if a vector is in the dual code using a generator matrix for $\mathcal{C}$.

Lemma 5.5. . *Let $\mathcal{C}$ be an $[n,k]_q$-linear code with generator matrix G. Then $\mathbf{v} \in \mathcal{C}^\perp$ if and only if $G\mathbf{v}^T = 0$.*

Proof. Let $\mathbf{r}_i \in \mathcal{C}$, $1 \leq i \leq k$ be the basis vectors of the code in the generator matrix $G \in M_{k,n}(\mathbb{F}_q)$. That is,

$$G = \begin{pmatrix} \leftarrow \mathbf{r}_1 \rightarrow \\ \vdots \\ \leftarrow \mathbf{r}_k \rightarrow \end{pmatrix}.$$

Now if $\mathbf{v} \in \mathcal{C}^\perp$, then by definition $\mathbf{r}_i\mathbf{v}^T = 0$. Hence

$$G\mathbf{v}^T = \begin{pmatrix} \leftarrow \mathbf{r}_1 \rightarrow \\ \vdots \\ \leftarrow \mathbf{r}_k \rightarrow \end{pmatrix} \mathbf{v}^T = \begin{pmatrix} \mathbf{r}_1\mathbf{v}^T \\ \vdots \\ \mathbf{r}_k\mathbf{v}^T \end{pmatrix} = \mathbf{0}. \tag{2}$$

To prove the reverse inclusion, assume that $G\mathbf{v}^T = 0$ and let $\mathbf{c} \in C$. As the $\mathbf{r}_i$ form a basis of $\mathcal{C}$, there exist $\lambda_i \in \mathbb{F}_q$ such that $\mathbf{c} = \sum_{i=1}^k \lambda_i \mathbf{r}_i$. But then

$$\mathbf{c}\mathbf{v}^\perp = \sum_{i=1}^k \lambda_i \mathbf{r}_i \mathbf{v}^T = 0$$

by (2). As $\mathbf{c}$ was chosen arbitrarily it follows that $\mathbf{v} \in \mathcal{C}^\perp$. □

Example 5.6. *Recall from example 3.17 that*

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

is a generator matrix for the simplex code $\mathcal{C}_3$. But we also saw that this was a choice of parity check matrix for the binary Hamming code $\mathcal{H}_3$! Hence $\mathbf{v} \in \mathcal{C}_3^\perp$ (for $\mathbf{v} \in \mathbb{F}_2^7$) if and only if $\mathbf{v} \in \mathcal{H}_3$, and so $\mathcal{C}_3^\perp = \mathcal{H}_3$ by Lemma 5.5.

We summarise the main properties of the dual code in the following theorem

Theorem 5.7. *Let $\mathcal{C}$ be an $[n,k]_q$-linear code.*

(1) Any generator matrix for $\mathcal{C}$ is a parity check matrix for $\mathcal{C}^\perp$. In particular $\mathcal{C}^\perp$ is an $[n, n-k]_q$-linear code;
(2) Any parity check matrix for $\mathcal{C}$ is a generator matrix for $\mathcal{C}^\perp$;
(3) $C = (\mathcal{C}^\perp)^\perp$.

Proof.

(1) The first part of this statement is exactly Lemma 5.5 and so it remains to confirm the parameters of $\mathcal{C}^\perp$. Well firstly, it is clear by definition that $\mathcal{C}^\perp \leq \mathbb{F}_q^n$. Now if G is a generator matrix of $\mathcal{C}$ then $\operatorname{rank}(G) = k$. Then as G is also a parity check matrix for $\mathcal{C}^\perp$, we have by the rank-nullity Theorem 2.14 that

$$\operatorname{rank}(\mathcal{C}^\perp) = \operatorname{nullity}(G) = n - \operatorname{rank}(G) = n - k.$$

(2) Let $H \in M_{n-k,n}(\mathbb{F}_q)$ be a parity check matrix for $\mathcal{C}$. It suffices to show that $\text{RowSpace}(H) = \mathcal{C}^\perp$. Now recall that $H\mathbf{c}^T = \mathbf{0}$ for all $\mathbf{c} \in \mathcal{C}$ by definition. So in particular, $\mathbf{r}_i\mathbf{c}^T = \mathbf{c}\mathbf{r}_i^T = 0$ where $\mathbf{r}_i$ denote the rows of H. Hence $\mathbf{r}_i \in \mathcal{C}^\perp$ for each i and we have the inclusion of subspaces

$$\langle \mathbf{r}_1, \dots, \mathbf{r}_{n-k} \rangle_{\mathbb{F}_q} = \text{RowSpace}(H) \leq \mathcal{C}^\perp.$$

But these two subspaces have the same dimension $n-k$ and are hence equal.

(3) Left as an exercise.

□

Corollary 5.8.

(1) Generator/parity check matrices for C correspond to parity check/ generator matrices for $\mathcal{C}^\perp$ respectively.

(2) Every linear code has a parity check matrix

Proof.

(1) This follows immediately from Theorem 5.7 (3).

(2) Let $\mathcal{C}$ be a linear code. Now very linear code has a basis and therefore a generator matrix. In particular, $\mathcal{C}^\perp$ has a generator matrix, say G. But then G is a parity check matrix for $\mathcal{C}$ by the first part of this corollary.

□

6. Perfect codes and other properties

The previous sections provided methods for constructing linear codes. Furthermore, we have developed techniques for calculating the parameters such as dimension and minimum distance. But which combinations of parameters are actually possible? There's no reason to believe that we can construct an $[n, k, d]_q$-linear code for any n, k, d, q of our choosing.

Ideally we want a code where we can send many messages i.e. $|\mathcal{C}|$ is large, as well as a high minimum distance $d(\mathcal{C})$ as this allows for greater error correction. But we also want to minimise the length n as this will affect the practical cost of transmitting codewords. A compromise needs to be found: in this section we'll see two important results concerning the existence and non-existence of linear codes with these desirable parameters.

Recall from Definition 3.5, that a sphere with centre $\mathbf{v} \in \mathbb{F}_q^n$ and radius $r \geq 0$ is the collection of vectors

$$S_r(\mathbf{v}) := \{\mathbf{u} \in \mathbb{F}_q^n \,:\, d(\mathbf{u}, \mathbf{v}) \leq r\}.$$

As the Hamming distance between two points is always a non-negative integer, it is enough to take the radius r to be an integer too. It is now easy to compute the size of these spheres.

Lemma 6.1. *Let $\mathbf{v} \in \mathbb{F}_q^n$ and $r \in \mathbb{N}$. Then*

$$|S_r(\mathbf{v})| = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

Proof. For each $0 \leq i \leq r$ let $S_i = \{\mathbf{u} \in \mathbb{F}_q^n \;\; d(\mathbf{u}, \mathbf{v}) = i\}$ i.e. the points that are Hamming distance exactly i from $\mathbf{v}$. Clearly $|S_r(\mathbf{v})| = |S_0| + |S_1| + \cdots + |S_r|$ and so it suffices to show that $|S_i| = \binom{n}{i}(q-1)^i$. Well, how many ways can we change exactly i entries of $\mathbf{v}$? Firstly we need to pick i of the co-ordinates to change: as there are n options, this gives $\binom{n}{i}$ choices. Then for each of position we wish to change, we have $|\mathbb{F}_q| - 1 = q - 1$ choices of field element. This gives $\binom{n}{i}(q-1)^i$ distinct choices in total, and the result follows.

□

Theorem 6.2. ***The sphere-packing bound.*** *Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a (not necessarily linear) code, with error correcting index e. Then*

$$|\mathcal{C}| \leq \frac{q^n}{\sum_{i=0}^{e} \binom{n}{i}(q-1)^i}.$$

In particular, if $\mathcal{C}$ is also an $[n,k]_q$-linear code then

$$1 \leq \frac{q^{n-k}}{\sum_{i=0}^{e} \binom{n}{i}(q-1)^i}.$$

Proof. As $\mathcal{C}$ corrects e errors, the spheres $S_e(\mathbf{c})$ are disjoint for all $\mathbf{c} \in \mathcal{C}$. It follows that

$$\begin{aligned} |\cup_{\mathbf{c}\in\mathcal{C}} S_e(\mathbf{c})| &= |\mathcal{C}||S_e(\mathbf{c})| \\ &= |\mathcal{C}|(\sum_{i=0}^{e} \binom{n}{i}(q-1)^i), \end{aligned}$$

by Lemma 6.1. But $\cup_{\mathbf{c}\in\mathcal{C}} S_e(\mathbf{c}) \leq \mathbb{F}_q^n$ and so

$$|\mathcal{C}| \sum_{i=0}^{e} \binom{n}{i}(q-1)^i \leq q^n.$$

□

Clearly a code that attains the sphere-packing bound is very useful, as it can send the largest number of codewords possible for a fixed n, q and e.

Definition 6.3. A code $\mathcal{C}$ that attains the sphere-packing bound is called *perfect*, or *e-perfect* if the error-correcting index is given.

We have already seen an infinite family of perfect codes

Example 6.4. *The Hamming codes $\mathcal{H}_k(q)$ are all 1-perfect.*

Proof. This is a straightforward computation: as $n = (q^k - 1)/(q-1)$ and $|\mathcal{H}_k| = q^{n-k}$, we simply need to check that

$$\sum_{i=0}^{e} \binom{n}{i}(q-1)^i = 1 + n(q-1) = 1 + \frac{q^k - 1}{q-1}(q-1) = q^k.$$

□

The sphere-packing bound is useful for showing the non-existence of codes:

Example 6.5. *Let $\mathcal{C}$ be a binary linear code of length 15 that corrects 2 errors. What is the maximum dimension of $\mathcal{C}$? Well, by Theorem 6.2*

$$|\mathcal{C}| \leq \frac{2^{15}}{1 + 15 + \binom{15}{2}} = 2^{15}/121 < 2^9.$$

Hence $|\mathcal{C}| \leq 2^8$ i.e. $\dim \mathcal{C} \leq 8$.

But this is not to say that such codes of dimensions $1, \ldots, 8$ all do actually exist!

We can get a bit closer to the truth using the following famous existence result for linear codes.

Theorem 6.6. ***The Gilbert-Varshamov bound****. Let n, k, d be positive integers, and q a prime power such that*

$$\sum_{j=0}^{d-2} \binom{n-1}{j} (q-1)^j < q^{n-k}. \tag{3}$$

Then there exists an $[n,k]_q$-linear code $\mathcal{C}$ with minimum distance $d(\mathcal{C}) \geq d$.

Proof. The proof of the G-V bound is constructive: first we assume that the positive integers n, k, d, q satisfy condition (3). Now to construct an $[n,k]_q$-linear code $\mathcal{C}$ of correct distance, it is sufficient by Proposition 4.4 to construct a parity check matrix H satisfying

(a) $H \in M_{n-k,n}(\mathbb{F}_q)$ of rank $n-k$;
(b) any $d-1$ columns of H are linearly independent.

We construct H inductively, column by column. Begin by choosing the first $n-k$ columns to be the standard basis vectors $\mathbf{e}_i^T$, $1 \leq i \leq n-k$. These are clearly all linearly independent.

Inductive step Suppose that we've chosen some i columns $\mathbf{h}_1^T, \ldots, \mathbf{h}_i^T \in \mathbb{F}_q^{n-k}$, such that $n-k \leq i \leq n-1$ and any collection of $d-1$ columns is linearly independent. Then

$$H_i = \begin{pmatrix} \uparrow & & \uparrow \\ \mathbf{h}_1^T & \cdots & \mathbf{h}_i^T \\ \downarrow & & \downarrow \end{pmatrix}$$

is $(n-k) \times i$ and satisfies (b). We now need to choose a further column $\mathbf{h}_{i+1} \in \mathbb{F}_q^{n-k}$ such that H_{i+1} still satisfies (b). Well how many choices of columns *won't* work?

We can't take any vector that is contained in the span of up to $d-2$ columns from $\{\mathbf{h}_1, \ldots, \mathbf{h}_i\}$. Consider a span of size $j \leq d-2$: firstly note that we can choose j distinct columns from $\{\mathbf{h}_1, \ldots, \mathbf{h}_i\}$ in $\binom{i}{j}$ ways. Furthermore, these columns span a set of size $(q-1)^j$ as we can pick any non-zero coefficient from $\mathbb{F}_q$ for each. It follows that the number of vectors that won't work is at most

$$1 + i \cdot (q-1) + \binom{i}{2}(q-1)^2 + \cdots + \binom{i}{d-2}(q-1)^{d-2}.$$

But since $i \leq n-1$, this number is less than q^{n-k} by the assumption in the statement of the Theorem. Hence there exists a vector that does work i.e. there exists $\mathbf{h}_{i+1} \in \mathbb{F}_q^{n-k}$ such that

$$H_{i+1} = \begin{pmatrix} \uparrow & & \uparrow \\ \mathbf{h}_1^T & \cdots & \mathbf{h}_{i+1}^T \\ \downarrow & & \downarrow \end{pmatrix}$$

does satisfy (b). This completes the inductive step, and we construct H_i for $i = n-k, \ldots, n$. The matrix $H = H_n$ is the final parity check matrix we require. □

Example 6.7. *Let's apply this to our problem of looking for codes in example 6.5: let $q = 2, n = 15$ and $d = 5$. We see that*

$$1 + 14 + \binom{14}{2} + \binom{14}{3} = 1 + 14 + 91 + 364 < 512 = 2^9 = 2^{15-6}.$$

Hence by Theorem 6.6 it follows that such a code (i.e. a binary code of length 15 and correcting 2 errors) of dimension 6 does exist. Unfortunately, neither the sphere-packing, nor G.V. bound tells us anything about the dimensions 7 and 8.

7. Equivalent codes and the weight enumerator

Properties such as dimension and minimum distance are useful for comparing two codes $\mathcal{C}_1$ and $\mathcal{C}_2$, especially when it comes to technological applications. However as mathematicians interested in classifying algebraic objects we may wish for a greater measure of whether $\mathcal{C}_1$ is "like" $\mathcal{C}_2$ - perhaps something resembling the property of two groups being isomorphic, or two matrices being conjugate. For this we introduce the notion of *equivalent* codes.

Firstly we need to define the action of the symmetric group, S_n, on a code $\mathcal{C} \subseteq \mathbb{F}_q^n$. Recall that S_n is simply the group of permutations on the letters $\{1, \ldots, n\}$.

Definition 7.1. Let $\mathcal{C} \leq \mathbb{F}_q^n$, $\mathbf{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ be a codeword, and let $\sigma \in S_n$. Then

$$\sigma(\mathbf{c}) := (c_{\sigma(1)}, \ldots, c_{\sigma(n)}).$$

We are now ready to define code equivalence.

Definition 7.2. Let $\mathcal{C}_1, \mathcal{C}_2$ be two $[n,k]_q$-linear codes. We say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *equivalent* if they differ only by a fixed reordering of the components of the codewords. More formally, we say that $\mathcal{C}_2$ is equivalent to $\mathcal{C}_1$ if there exists a fixed $\sigma \in S_n$ such that for all $\mathbf{c} \in \mathcal{C}_2$, there exists $\mathbf{d} \in \mathcal{C}_1$ such that $\sigma(\mathbf{d}) = \mathbf{c}$. If this holds, we denote equivalence by $\mathcal{C}_1 \cong \mathcal{C}_2$.

Example 7.3. *Let*

$$\mathcal{C}_1 := \{(0000), (0011), (1100), (1111)\} \subset \mathbb{F}_2^4;$$
$$\mathcal{C}_2 := \{(0000), (0101), (1010), (1111)\} \subset \mathbb{F}_2^4.$$

Clearly, $\mathcal{C}_1 \cong \mathcal{C}_2$ via the permutation $(2,3) \in S_4$.

We can check for code equivalence in a natural way using generator matrices.

Lemma 7.4. *Let $\mathcal{C}_1, \mathcal{C}_2 \leq \mathbb{F}_q^n$ be linear codes with generator matrices $G_1, G_2 \in M_{k,n}(\mathbb{F}_q)$, respectively. Suppose that there exists a $n \times n$ permutation matrix P (*) such that $G_1 P = G_2$. Then $\mathcal{C}_1 \cong \mathcal{C}_2$.*

Proof. Left as an exercise □

Remark. (*) Recall that a permutation matrix is an invertible square matrix such that each row and each column contains exactly one entry 1, with all other entries equal to 0. Clearly there is a bijection between such matrices and the elements of S_n.

Exercise 7.5. *Recall from Definition 4.1 that there is more than one binary Hamming code $\mathcal{H}_k$ for a fixed k- they correspond to different choices of parity check matrix. Show that all of these Hamming codes are equivalent.*

In addition to the minimum distance of a linear code, we may also consider finer structure it possesses. For example, given a code $\mathcal{C}$ it is natural to ask how many codewords there are of weight m?

We create a "book-keeping device" that keeps track of these quantities called the *weight enumerator*.

Definition 7.6. Let $\mathcal{C} \leq \mathbb{F}_q^n$ be a linear code. The *weight enumerator*, $W_{\mathcal{C}}(x,y)$ is the polynomial

$$W_{\mathcal{C}}(x,y) := \sum_{\mathbf{c} \in \mathcal{C}} x^{n - \mathrm{wt}(\mathbf{c})} y^{\mathrm{wt}(\mathbf{c})} = \sum_{m=0}^{n} A_m x^{n-m} y^m,$$

where $A_m = |\{\mathbf{c} \in \mathcal{C} \mid \mathrm{wt}(\mathbf{c}) = m\}|$.

Example 7.7.

(1) The n-bit repetition code $\mathcal{C} = \mathcal{R}_n$ over $\mathbb{F}_2$ has $W_{\mathcal{C}}(x,y) = x^n + y^n$.

(2) *Consider the simplex code $\mathcal{C}_3$ of length 7 and dimension 3 (see example 1.2). It is easy to see from the natural generator matrix that each non-zero codeword has weight four. Hence $W_{\mathcal{C}_3}(x,y) = x^7 + 7x^3y^4$.*

We deduce the following properties of the weight enumerator.

Lemma 7.8. *Let $W_{\mathcal{C}}(x,y)$ be the weight numerator for $\mathcal{C}$, as above. The following hold*

(1) $W_{\mathcal{C}}(1,1) = \sum_{m=0}^{n} A_m = |\mathcal{C}|$;
(2) $d(\mathcal{C}) = \min(\{m \neq 0 \,|\, A_m \neq 0\})$.

Proof. The proof of both statements follow easily from the definition. □

Lemma 7.9. *Let $\mathcal{C} \leq \mathbb{F}_2^n$ be a binary linear code. Then $W_{\mathcal{C}}(x,y) = W_{\mathcal{C}}(y,x)$ if and only if $\mathbf{1} = (1,\ldots,1) \in \mathcal{C}$.*

Proof. As $\mathcal{C}$ is linear, $\mathbf{0} \in \mathcal{C}$, and therefore $W_{\mathcal{C}}(x,y)$ contains the term x^n. So assuming $W_{\mathcal{C}}(x,y) = W_{\mathcal{C}}(y,x)$, it follows that the weight enumerator contains the term y^n and by definition $\mathbf{1} \in \mathcal{C}$. For the reverse inclusion, notice that the map $f : \mathcal{C} \to \mathcal{C}$ given by $f(\mathbf{c}) = \mathbf{c} + \mathbf{1}$ is a bijection that maps codewords of weight i to codewords of weight $n - i$. Hence if $\mathbf{1} \in \mathcal{C}$ then

$$|\{\mathbf{c} \in \mathcal{C} \,|\, \mathrm{wt}(\mathbf{c}) = i\}| = |\{\mathbf{c} \in \mathcal{C} \,|\, \mathrm{wt}(\mathbf{c}) = n - i\}|$$

for all $0 \leq i \leq n$. The weight enumerator is then symmetric by definition. □

Lemma 7.10. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be equivalent linear codes. Then $W_{\mathcal{C}_1}(x,y) = W_{\mathcal{C}_2}(x,y)$.*

Proof. This follows easily from the definition of equivalence once we notice that the permutation action preserves codeword weight. □

So we have (an incredibly strong!) relationship between the weight enumerators of equivalent codes. Can we also find a relationship concerning the enumerators of dual codes? When the codes are binary, we have the following well-known and elegant result.

Theorem 7.11. ***MacWilliams Identity:*** *Let $\mathcal{C}$ be a $[n,k]_2$-code. Then*

$$W_{\mathcal{C}^\perp}(x,y) = \frac{1}{2^k} W_{\mathcal{C}}(x+y, x-y).$$

Example 7.12. *Recall that the simplex code $\mathcal{C}_3$ is dual to $\mathcal{H}_3$, the binary hamming code of length 7. Hence*

$$\begin{aligned} W_{\mathcal{H}_3} &= \frac{1}{2^3} W_{\mathcal{C}_3}(x+y, x-y) \\ &= \frac{1}{2^3}(x+y)^7 + \frac{1}{2^3}(x+y)^3(x-y)^4 \\ &= x^7 + 7x^4y^3 + 7x^3y^4 + y^7. \end{aligned}$$

Example 7.13. *More generally, consider $\mathcal{H}_k$, the binary Hamming code of length $n = 2^k - 1$. Here the weight enumerator has the form*

$$W_{\mathcal{H}_k}(x,y) = \frac{x^n}{n+1}(1 + \frac{y}{x})^n + \frac{nx^n}{n+1}(1 + \frac{y}{x})^{(n-1)/2}(1 - \frac{y}{x})^{(n+1)/2}. \tag{4}$$

The proof of (4) is beyond the scope of this course (it involves solving a differential equation!) but if you are interested, the details can be found in Van Lint (page 41).

Let's compute the weight enumerator for $\mathcal{H}_4$: well $n = 15$ and hence

$$\begin{aligned} W_{\mathcal{H}_4}(x,y) &= \frac{x^{15}}{15+1}(1+\frac{y}{x})^{15} + \frac{15x^{15}}{15+1}(1+\frac{y}{x})^7(1-\frac{y}{x})^8 \\ &= x^{15} + 35x^{12}y^3 + 105x^{11}y^4 + 168x^{10}y^5 + 280x^9y^6 + 435x^8y^7 \\ &+ 435x^7y^8 + 280x^6y^9 + 168x^5y^{10} + 105x^4y^{11} + 35x^3y^{12} + 1. \end{aligned}$$

Notice that $W_{\mathcal{H}_3}$ and $W_{\mathcal{H}_4}$ are symmetric - that is they are fixed under exchanging the variable x and y.

Exercise 7.14. *Prove this holds in general i.e. for $\mathcal{H}_k$, $k \geq 3$.*

Alexander J. Malcolm, School of Mathematics, University of Bristol, Bristol, BS8 1TW, UK, and the Heilbronn Institute for Mathematical Research, Bristol, UK.
E-mail address: alex.malcolm@bristol.ac.uk