Coding Theory

Paul Martin

September 28, 2019

School of Maths

Coding theory

Hello, everyone!

Contents

Coding Theory

Preamble

What? Why?

The aim of this chapter is to study coding theory. We begin with a few general words about what coding theory *is*, and why we want to study it (i.e. what is it good for?). [1]

[1] Warning: This preamble is mildly philosophical in nature. It might be best to skip it for now, and come back after you have studied a few chapters of coding practice.

Coding

Coding is the act of preparing information for transmission.[2]

[2] For example, from Dictionary.com:
`http://dictionary.reference.com/browse/coding` we have:
11. Linguistics. a. the system of rules shared by the participants in an act of communication, making possible the transmission and interpretation of messages.

There are many subtleties to this definition. For example, in order to verify that information exists it has to be transmitted, so coding is effectively part of the creation of information. Anyway, from the pseudo-definition above it will already be clear that coding is 'important'. It also contains language as a substructure, which further emphasises its importance.[3]

[3] See for example
http://leoalmanac.org/journal/Vol_14/lea_v14_n05-06/lglazier.asp

All transmission carries the risk of corruption. The *Science* (or Theory) of coding is concerned with minimising this risk (in some, usually quantitatively probabilistic, sense).

Example: suppose we need to be certain a message has got through exactly as sent (e.g. a 'zipped tar' file). [4] What can we do?

[4] What does 'certain' mean here?! This is another Statistics-meets-Physics/Philosophy question...

.

As you can see, intrinsic to this chapter are notions of communication, data, risk, and information. These are not trivial notions, and we won't be able to define them up-front. In mathematics we try to keep the number of terms that are used without definition to a minimum. This is because every term used without definition is a possible source of confusion between person A (the propagator, perhaps, of an idea) and person B (the recipient). Person A simply has to trust that person B is understanding the same thing by her term. If not, then any idea built on it will be flawed. Unfortunately it is never possible to define *all* terms. In mathematics, for example, we generally take on trust that others understand the same thing by the term 'set' as we do.

In the *applications* of mathematics, however, this 'define everything' discipline can conflict with progress. Our strategy will be to use some terms, where necessary, without an initial definition; but to try to come back to them later and find a way to check that we really do agree on their meaning.

.

To begin with, then, we may consider *communication* as the process whereby some 'data' held in one 'machine' is passed so that a copy is held in some other machine. This is somewhat analogous to the process whereby an idea held in your mind might be communicated to me, so that I then hold that same idea. [5] The extent to which this analogy works (or, if failing, we still want to treat both processes) is a matter for discussion. It is probably true to say that we can work more comfortably with the first process than the second, but the second is ultimately perhaps more interesting?

[5] Descartes doubts even that other people exist, so communicating with them is something not to be taken lightly, if we are being really careful! We simply can't afford to be this careful here — we have concrete applications to address.

.

I thank Martin Speight for lending me his own beautiful notes on Coding Theory, which have been invaluable in the preparation of this Chapter.

Some recommended reading:

R Hill, "First Course in Coding Theory", Clarendon Press, 1986.

G A Jones and J M Jones, "Information and Coding Theory", Springer, 2000.

...In a single picture

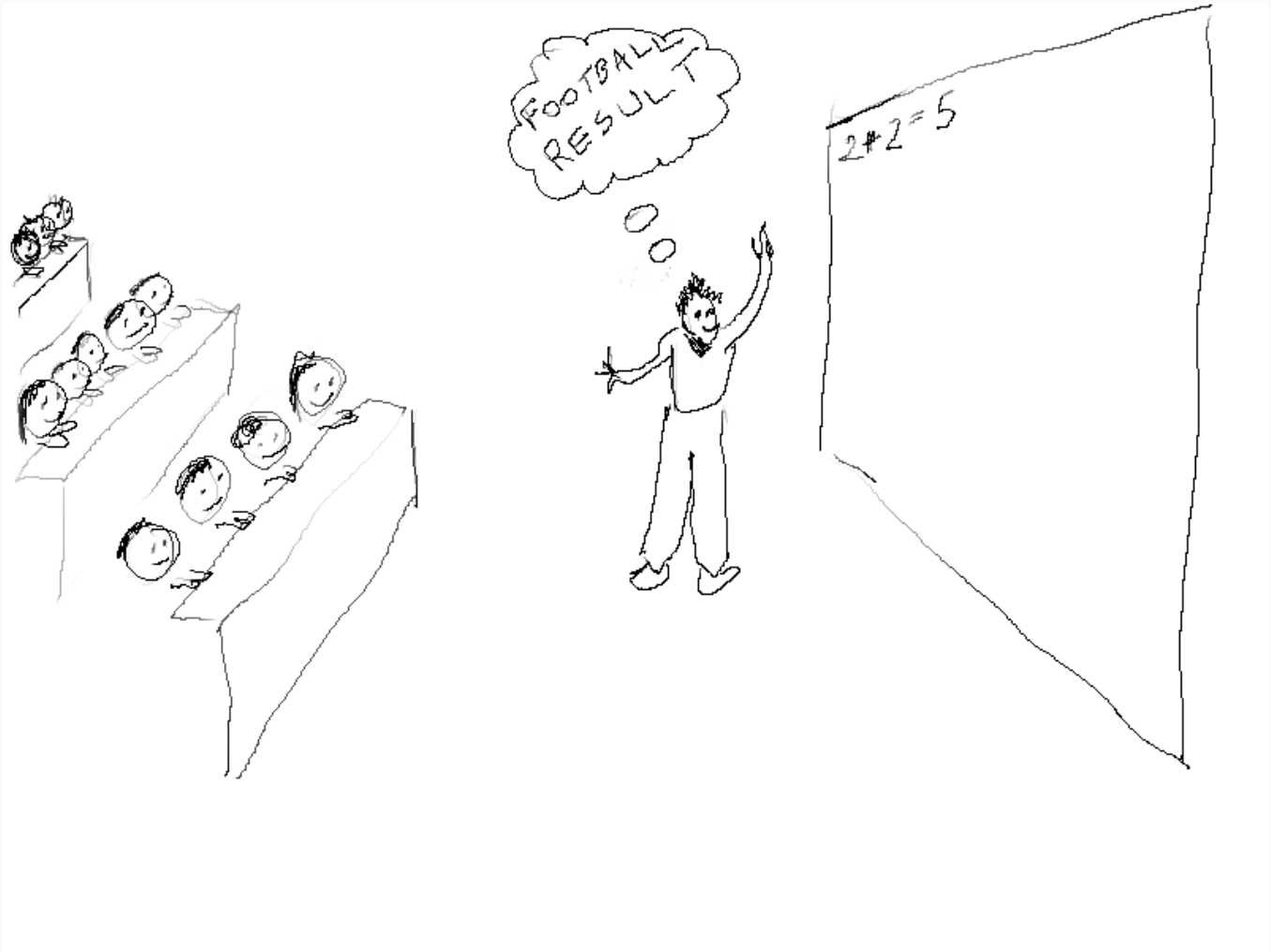


Figure 1: A coding theory class.

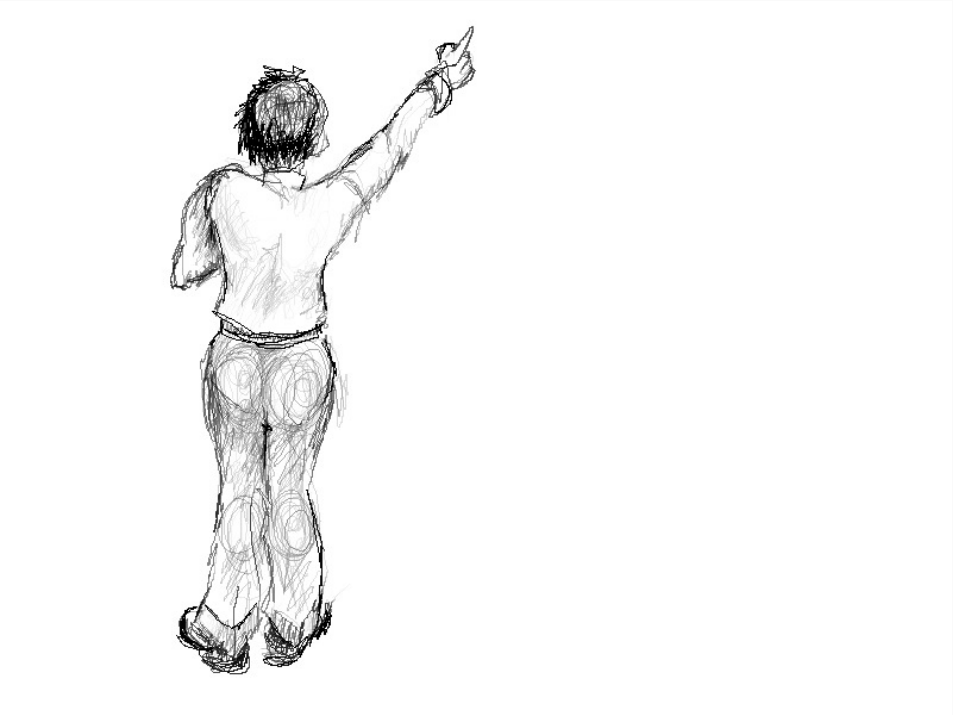


Figure 2: A noisy channel.

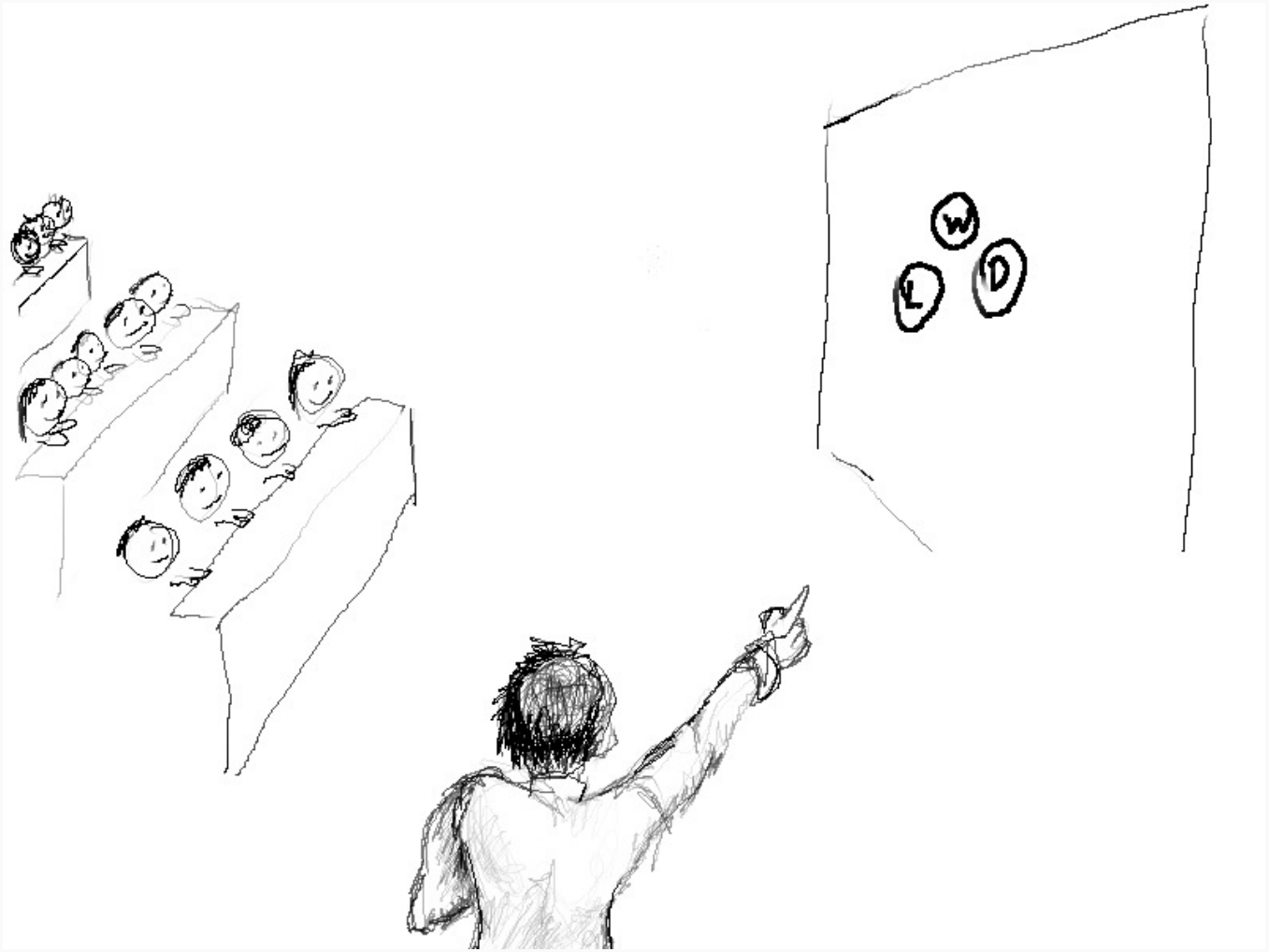


Figure 3: Transmitting data through a noisy channel.

First you need to look at figure 1. But then... See figure 3. Here person **A** tries to communicate the result of a football match (Win, Lose or Draw). This is done by: (1) setting up an 'encoding' of the set of possible messages (W,L,D) — in this case by associating them with different points on the whiteboard; (2) transmitting the match result down a noisy channel — in this case by pointing at it. (This communication method might not be a very *good* practical communication method under the circumstances, but it contains nice analogies of many of the key points of coding theory.) All these ideas will be explained as we go along.

Coding

Definitions

A coding is a representation of data. (What is data?...)

Let S be a set. A sequence of elements of S of length l is an element of

$$S^l = S \times S \times ... \times S$$

For example, if $S = S_{alph}$ is the usual 26 element alphabet then

$$(w, i, l, l, y, o, u, m, a, r, r, y, m, e)$$

is a sequence of length 14. (Where no ambiguity arises we might drop the brackets and commas.)
A *finite* sequence is a sequence of finite length.

Definitions

Define S^0 to be (the set containing) the empty sequence, and

$$S^* = \cup_{l \geq 0} S^l \qquad \text{and} \qquad S^+ = \cup_{l > 0} S^l$$

Define a product on S^* by

$$\begin{array}{rcl} \circ : S^* \times S^* & \to & S^* \\ (x, y) & \mapsto & x \circ y = xy \end{array}$$

where xy is the concatenation of x and y.

0.1. Example. If $x = 01010110$ and $y = 1$ then $xy = x1 = 010101101$.

Wisdom

A significant percentage of all human wisdom (?!), and all human communication, has been encoded as sequences using a mild generalisation of the alphabet S_{alph}. [6]

[6] On the other hand there is no system which will enable us to encode even a single 'generic' element of the set $(0, 1)$ (the unit open real interval).
Some elements in this interval can be communicated by more abstract means. For example π, e and $\sqrt{2}$. Such abstractions are presently among the features distinguishing humans from computers... but that is another story.

Data

'Data', for our present purposes, takes the form of some finite sequence. We assume that this sequence has value to us for some reason (determining the humanistic value of a given sequence is beyond the scope of this section, but it might contain, for example, a list of transactions in your bank account for the last year). The challenge we face is to transmit this data to a new location.
For example, perhaps a person in England wants to communicate the question implied by the sequence
$(w, i, l, l, y, o, u, m, a, r, r, y, m, e)$ to a friend in Australia. In this case obviously shouting it out, or writing it onto a sheet of paper and throwing this in a southerly direction, is not going to get the job done, even if the recipient knows to expect a message (audible or written, respectively) in some given time-window. Phoning or sending an email might work better. But all these efforts can be considered as involving the same basic process:

The basics

1. Source (person A) has a message to communicate (I want to offer marriage to person B). We shall assume that the source has this message expressed as a finite sequence in some source alphabet T.

2. Source message is encoded in some way suitable for travel to B (for example by vocalising in spoken English — whatever that is). We shall assume that the encoding passes the message to a sequence in a not necessarily distinct coding alphabet S.

3. Encoded version travels somehow to B, degrading gradually for various reasons as it travels;

4. Degraded encoded version reaches target's decoder (nominally in our example it is a sound, so the decoder is an ear/brain system; but obviously the sound heard by B at the appropriate point in time will have only a negligible amount of correlation with the original encoding). An attempt is made to decode this version.

5. Some approximation to the original message arrives for use at the target.

A code

0.2. A code C for a source alphabet T is a function $f : T \to S^+$ to sequences in code alphabet S. The properties of codes that we shall focus on depend on the image set $f(T)$ rather than the details of the map itself, so one often regards a code simply as this set of words.

The *extension* of C to T^* is obtained simply by using f to encode each symbol in succession.

0.3. Example. (i) If $f(a) = 001$ and $f(b) = 010$ then $f(abba) = 001010010001$.
(ii) If $f(a) = 1$ and $f(b) = 010$ then $f(abba) = 10100101$.

.

We shall be interested in fixed-length codes (a discussion of variable length codes can be found for example in Jones and Jones (Springer SUMS, 2000)):

0.4. Definition. A block code

$$C = \{(x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n), ...\}$$

of length n over set S is a subset of S^n. Code C is q-ary if $|S| = q$.

Encoding

An encoding is a recasting of one code as another (or the encoding of a message, but no usable message is really entirely unencoded).

0.5. Example. Let S, T be sets and

$$f : T \to S^l$$

Then a code $C \subset T^n$ can be coded over S by applying f to each element of each sequence x in turn as before. This time:

$$f : T^n \to S^{nl}$$

where $f(x)_{(i-1)l+(j-1)+1} = f(x_i)_j$ for $j - 1 < l$.

.

In particular (1) if $T = \{N, S, E, W\}$ and $S = \{0, 1\}$ and $f_1(N) = (0, 0) = 00$, ..., $f_1(W) = (1, 1) = 11$ then

$$f_1(EESW) = f((E, E, S, W)) = 10100111$$

(2) if T, S as above and $f_2(N) = 000$, $f_2(S) = 011$, $f_2(E) = 101$, $f_2(W) = 110$, then

$$f_2(EESW) = 101101011110$$

Transmission

Transmit

Now suppose we transmit the message $EESW$ — in any invertible encoding.

We assume that the recipient knows (1) that the original message was some sequence in $\{N, S, E, W\}$, and (2) how we encoded it (if at all).

Thus, if the encoded message arrives intact, she can invert the encoding to recover the original message.

BUT We want to consider the realistic scenario in which, with some probability, part of the encoded message is corrupted in transmission.

We want to ask: What can be done about that? And what can 'best' be done?

.

For example, suppose that there is a 1% chance that recipient B will mishear any term in the sequence in the original encoding. Then there is a roughly 4% chance that the message will arrive with a corrupted element.

Note that there is no way for the recipient to tell whether the message has been corrupted or not, in the original encoding or in f_1 (from Example 0.5).
In f_2, however, not every binary code of length 3 is the image of an element of T, so if 101 was corrupted to 001, say, we would know at least that there had been an error in transmission. Indeed with this encoding *every* single element transmission error would show up. However double errors could still appear to be OK.

Now consider
(3): T, S as in Example 0.5 above and $f_3(N) = 00000$,
$f_3(S) = 01101$, $f_3(E) = 10110$, $f_3(W) = 11011$.

0.6. Exercise. Verify that if any two errors occur then the received message is not the image of any sent message, signaling an error.

Further, if a single error occurs the sent message is recoverable anyway. For example suppose $E \mapsto 10110 \to 10010$ after transmission. We cannot decode this, but considering the following table of number of places differing from the encoding of each element of T:

.

encoding	places differing
00000	2
01101	5
10110	1
11011	2

we guess correctly that the intended element was E.
We say that (3) is 2 error *detecting*; or single error *correcting*.

Note that the cost of these improvements was higher block length, which introduced some redundancy. That is, we have a trade-off between efficiency and reliability.

Hamming distance

Metric

Let us try to be more precise about this error-correcting facility. Recall

0.7. Definition. Let S be a set. A map $d : S \times S \to \mathbb{R}$ is a metric on S if it satisfies: (i) $d(x, y) = 0 \iff x = y$
(ii) $d(x, y) = d(y, x) \forall x, y \in S$
(iii) $d(x, y) \leq d(x, z) + d(z, y) \forall x, y, z \in S$ (triangle inequality).

Note that the usual distance in Euclidean space $\mathbb{R}^n$ is a metric.

.

We don't have numbers (necessarily) in our 'alphabets', so our basic distance function is cruder:

0.8. Definition. Given $x, y \in S^n$ the (Hamming) distance between them is $d(x, y) =$ number of positions in which x, y differ.

0.9. Proposition. *The Hamming distance is a metric.*

Prove it!

Minimum distance

0.10. Definition. The minimum distance of a code $C \subset S^n$ is

$$d(C) = \min\{d(x, y) | x, y \in C, x \neq y\}$$

Examples:

C_1	00	01	10	11
00		1	1	2
01			2	1
10				1
11				

so that $d(C) = 1$ in case (1). Similarly in case (2) above the min distance is 2; and in case (3) it is 3. (Exercises!)

.

0.11. Proposition. *(a) If $d(C) \geq t+1$ then C can detect up to t errors;*
(b) If $d(C) \geq 2t+1$ then C can correct up to t errors by the 'pick the closest' strategy.

Proof: Exercise, or see below.

Balls and spheres

0.12. Definition. For any $x \in S^n$ and $r \in \mathbb{N}$ the ball of radius r (or r-ball) centred on x is

$$B_r(x) := \{y \in S^n | d(x, y) \leq r\}$$

That is, the set of sequences that differ from x in no more than r places.
An r-sphere is

$$S_r(x) := \{y \in S^n | d(x, y) = r\}$$

That is, the 'outer shell' of an r-ball.

0.13. Let $C \subset S^n$. Consider the collection of t-balls centred on all $x \in C$. This is a 'fuzzy picture' of the elements x. Each is surrounded by the area of uncertainty in it, in a neighbourhood of S, caused by up to t transmission errors.

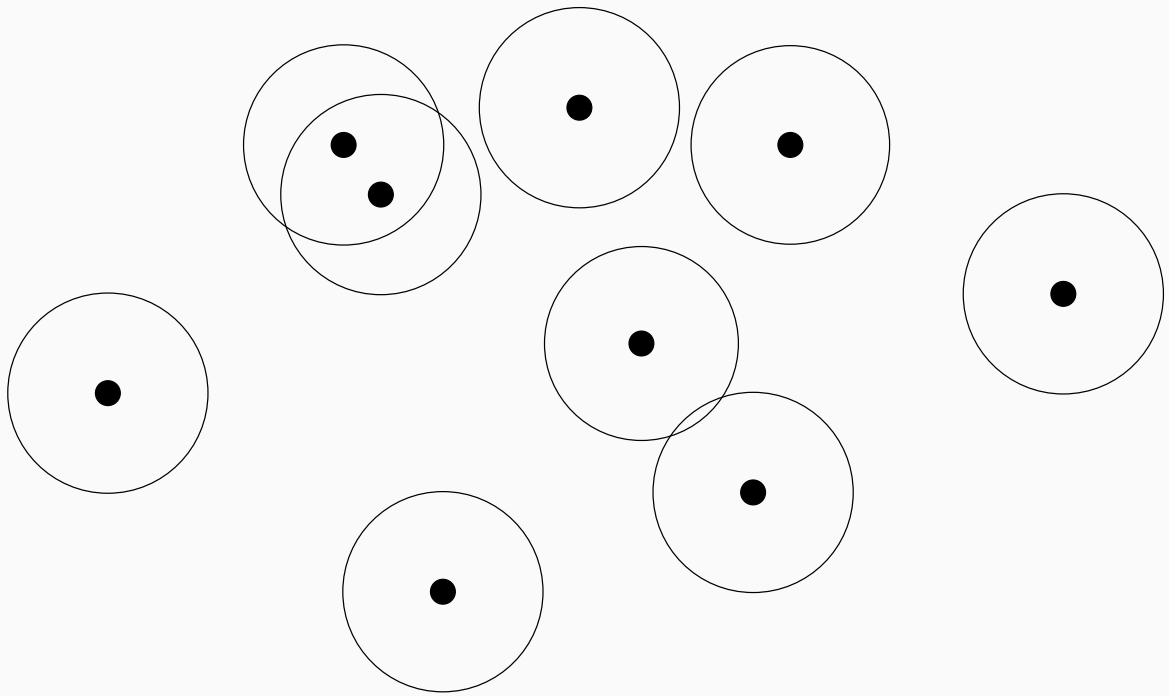


Figure 4: Ball packing heuristic (using Euclidean metric).

(a) If $d(C) \geq t+1$ then no x lies in another's ball. Thus if 1 up to t errors occur then the received message is not in C and we *know* we have an error.
(b) If $d(C) \geq 2t+1$ then even the balls are disjoint (this is perhaps not so obvious with the Hamming distance, cf. say the usual Euclidean metric, but the triangle inequality is what we need to confirm it), and if 1 up to t errors occur then the received message is closer to x than any other $y \in C$. $\square$

Optimisation

Code choice affects transmission error probability

We are making a case, superficially, that code f_3 is more reliable than f_1 when transmitting over a channel with errors. But in replacing by sequences 2.5 times as long we are giving it far more digits to get wrong! Is f_3 really more reliable? Less reliable? Does it really make any difference?

To settle this we need to compute a probability for a message being wrongly decoded in each case.
In order to do this it is simplest to make some assumptions about error probabilities in the transmission 'channel':
(a) Each transmitted digit is equally likely to be corrupted, with probability p.
(b) If a digit is corrupted, any of the $q-1$ other letters in S are equally likely to occur.

This is a q-ary symmetric channel, with symbol error probability p.

.

Sending symbol S in the f_1 code we send 01. It will be decoded correctly *only* if no errors occur:

$$P_{corr}(01) = (1-p)^2$$

so the error probability is

$$P_{err}(01) = 1 - (1-p)^2 \tag{1}$$

In f_3 we send 01101. This will decode correctly if 0 or 1 errors occur (possibly more) so

$$P_{corr}(01101) \geq (1-p)^5 + 5p(1-p)^4$$

so

$$P_{err}(01101) = 1 - (1-p)^5 - 5p(1-p)^4 \tag{2}$$

If p is small then (2) is much smaller than (1). E.g. if $p = 0.01$ then $P_{err}(01) = .0199$ while $P_{err}(01101) \leq .0009801496$. So increasing word length by 2.5 times reduced error probability 20-fold!

If p is bigger then f_1 doesn't look so bad (for example at around $p = .4$ and above it is better than f_3).

Anyway, the point is it makes a difference. So the science of coding theory is non-trivial. The game is ON!

0.14. Definition. A q-ary (n, M, d)-code is a block length n code with M codewords and minimum distance d.

For S a set let $P(S)$ denote the power set of S. Thus $P(S^n)$ is the set of length-n $|S|$-ary codes; and a q-ary (n, M, d)-code C is an element of $P(S^n)$ (some S of degree q) such that $|C| = M$ and $d(C) = d$.

As a convention, by default we assume that if $|S| = q$ then

$$S = \{0, 1, ..., q - 1\}$$

Write (n, M, d)-cod_q for the set of q-ary (n, M, d)-codes (or just (n, M, d)-cod if q is fixed). Thus:

$$P(S^n) = \sqcup_{M,d}(n, M, d)\text{-cod}$$

The A-function

Define

$$A_q(n, d) = \max M, \text{ for fixed } q, n, d$$

that is, the size of the largest possible q-ary (n, M, d)-code. Since q, n determine the size of the 'space' in the picture we considered earlier, and d the size of the 'exclusion zone' around each point — a ball in that space, it is reasonable that only so many such balls can be fitted in the space without overlap.

The following gives an upper bound on $A_q(n, d)$:

Theorem 0.15. *(Singelton bound) For any q-ary (n, M, d)-code, $M \leq q^{n-(d-1)}$. Hence $A_q(n, d) \leq q^{n-(d-1)}$.*

Proof: Let C be such a code, with code alphabet S, and $\pi : C \to S^{n-(d-1)}$ be the map

$$\pi : (x_1, x_2, .., x_n) = (x_1, x_2, .., x_{n-(d-1)})$$

Take $x \neq y \in C$. If $\pi(x) = \pi(y)$ then x, y agree in $n - (d - 1)$ places and hence differ in at most $d - 1$. But then $d(x, y) \leq d - 1$. Hence π is one-to-one. Hence its domain is no larger than its codomain:

$$M = |C| \leq |S^{n-(d-1)}| \; = \; q^{n-(d-1)}$$

□

Example

This singleton bound is not usually a very good bound, but is saturated in some circumstances.

0.16. Example. What is $A_2(3,2)$? By the singleton bound

$$A_2(3,2) \leq 2^{3-(2-1)} = 2^2 = 4$$

But our example (2) is a 2-ary (3,4,2)-code, so $A_2(3,2) \geq 4$. Hence $A_2(3,2) = 4$.

A much better upper bound is generally given by the 'ball packing argument'. This is built on a consideration of the amount of 'space' occupied by the 'error ball' around a codeword transmitted with a given number of errors:

0.17. Lemma. *If $x \in S^n$ then*

$$|B_t(x)| = \sum_{r=0}^{t} \binom{n}{r} (q-1)^r$$

Proof: $|S_r(x)|$ is the number of strings in S^n differing from x in precisely r places. This is product of the number of ways to pick the r differing places with the number of ways to assign a differing digit in each place:

$$|S_r(x)| = \binom{n}{r} (q-1)^r$$

□

Theorem 0.18. *(Ball packing bound) Let C be a q-ary (n, M, d)-code with $d \geq 2t + 1$. Then*

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r \leq q^n$$

Proof: Since $d \geq 2t + 1$, the t-balls centred on codewords are all disjoint. Hence

$$|\cup_{x \in C} B_t(x)| = \sum_{x \in C} |B_t(x)| = M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r$$

by Lemma 0.17. But

$$(\cup_{x \in C} B_t(x)) \subset S^n \Rightarrow |\cup_{x \in C} B_t(x)| \leq |S^n| = q^n$$

□

Using the BP bound

We can use this bound to rule out the existence of codes with certain properties. For example, there is no 3-ary (6,10,5)-code, since, with $t = 2$ ($d = 2 \times 2 + 1$)

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r = 730$$

while $q^n = 3^6 = 729$.

.

However, even if $q, (n, M, d)$ passes the BP bound it does not *follow* that a code exists. For example, there is no 2-ary (6,9,4)-code, even though

$$M \sum_{r=0}^{t} \binom{n}{r} (q-1)^r = 9(1+6) = 63 < 64 = q^n$$

In this case we can actually rule out a code using the singleton bound:

$$q^{n-(d-1)} = 2^{6-3} = 8$$

while $M = 9$. But even if $q, (n, M, d)$ passes both bounds it does not follow that such a code exists. (See table 1 for example.)

n	$d=3$			$d=5$		
	actual	*singleton*	*ball−packing*	*actual*	*singleton*	*bp*
5	4	8	5	$2*$	2	$2*$
6	8	16	9	$2*$	4	$2*$
7	$16*$	32	$16*$	2	8	4
8	20	64	28	4	16	6
9	40	128	51	6	32	11
10	$72-79$	256	93	12	64	18
11	144	512				
12	256	1024				
13	512	2048				
14	1024	4096	1092	128	1024	154
15	$2048*$	8192	$2048*$	256	2048	270
16	$2560-3276$	16384	3855	$256-340$	4096	478
17	$\geq 83*2^6$					
$\vdots$						
47	$\geq 9*2^{48}$					
$\vdots$						
163	$\geq 19*2^{151}$					

Table 1: Table of known values for $A_2(n,d)$, and some bounds. (See R Hill, A First Course in Coding Theory; or N Sloane's online page: http://www.research.att.com/∼njas/codes/And/. The most recently discovered of the entries given here is from around 1995.).

.

Which value of t do we use in the BP bound? The largest t such that $2t + 1 \leq d$, that is, $t \leq (d-1)/2$.

The largest integer not exceeding $z \in \mathbb{R}$ is written $\lfloor z \rfloor$ ('Floor function'). So use

$$t = \lfloor \frac{1}{2}(d-1) \rfloor$$

So the BP theorem implies

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

since $A_q(n,d)$ is an integer by definition. (These are the values tabulated under ball-packing.)

Perfect codes

Note that the collection of t-balls is disjoint. If they completely cover S^n this is obviously the best use of the 'space' we can make, and the code is said to be *perfect*.

0.19. Definition. A q-ary (n, M, d)-code is perfect if the collection of t-balls centred on codewords, $t = \lfloor (d-1)/2 \rfloor$, is a partition of S^n.

Note that this happens if and only if equality occurs in Theorem 0.18.

Note also that this cannot happen if d is even (exercise).

Example

0.20. Example. For our existing examples:

(1) is trivially perfect.

(2) $d = 2$ is even, so not perfect.

(3) is a 2-ary (5,4,3)-code:

$$M \sum_r \binom{n}{r} (q-1)^r = 4(1 + \binom{5}{1} 1) = 24$$

while $|S^5| = 2^5 = 32$, so not perfect.

Repetition code

0.21. Another kind of error-robust code, favoured by deaf people such as the author (!)[7], is a repetition code. A binary repetition code of length $n = 2t + 1$ is

$$C = \{00...0, 11...1\}$$

Clearly this is a $(2t + 1, 2, 2t + 1)$-code.
Every string $y \in S^{2t+1}$ either has more 0s than 1s, implying $y \in B_t(00...0)$;
or more 1s than 0s, implying $y \in B_t(11...1)$.
Hence $S^{2t+1} = B_t(00...0) \sqcup B_t(11...1)$.

[7] and roadies

0.22. Now, why did we only include the d odd cases in our table of $A_2(n, d)$?

For $A_2(n, d)$ we can deduce the even d cases from the odd.

0.23. Definition. The weight of a string $x \in S^n$ is

$$w(x) = \#\text{non-zero entries in } x$$

E.g. $w(011) = 2 = w(10010)$.

0.24. Lemma. *Suppose $S = \{0, 1\}$ and $x, y \in S^n$ both have even weight. Then $d(x, y)$ is even.*

Proof: Let $\underline{n} = \{1, 2, ..., n\}$ and, fixing x, y,

$$\underline{n}_{ij} = \underline{n}_{ij}(x, y) = \{k \in \underline{n} \mid x_k = i \text{ and } y_k = j\}$$

For example if $x = 01101$, $y = 10110$ then $\underline{n}_{00} = \emptyset$ and $\underline{n}_{01} = \{1, 4\}$.

(We will give the proof in the binary case as stated. Generalisations of the result are possible. Formulation of a suitable statement is left as an exercise (but will not be needed here).)
Now $w(x) = |\underline{n}_{10}| + |\underline{n}_{11}| = 2l$ for some l, since $w(x)$ is even; and $w(y) = |\underline{n}_{01}| + |\underline{n}_{11}| = 2m$ for some m similarly. Thus

$$d(x, y) = |\underline{n}_{10}| + |\underline{n}_{01}| = 2l + 2m - 2|\underline{n}_{11}|$$

□

0.25. Definition. An q-ary (n, M, d)-code is optimal if $M = A_q(n, d)$.

For $k \in \{1, 2, ..., n\}$ define 'projection'

$$\pi_k : S^n \to S^{n-1}$$

by $x \mapsto \pi_k(x) = x_1 x_2 ... x_{k-1} x_{k+1} ... x_n$ (deleting the k-th digit). This also acts, by restriction, on any subset of S^n, and hence on any code $C \in P(S^n)$, to produce a new code $\pi_k(C) \in P(S^{n-1})$.

For $i \in S$ define 'projection onto $x_k = i$-hyperplane' (abusing notation as if S^n were $\mathbb{R}^n$)

$$\pi_k^i : S^n \to S^n$$

by $x \mapsto \pi_k(x) = x_1 x_2 ... x_{k-1} i x_{k+1} ... x_n$ (replacing the k-th digit by i).

Using π_k

Note that if $D \in (n, M, d)$-cod with $d > 1$ then $|\pi_k(D)| = M$, since the maximum reduction in distance between distinct points caused by deleting one letter is 1 (so distinct points are still distinct after projection). That is

$$\pi_k : (n, M, d+1)\text{-cod} \to \sqcup_{d' \in \{d, d+1\}} (n-1, M, d')\text{-cod}$$

Theorem 0.26. *Suppose d odd. A 2-ary (n, M, d)-code exists iff a 2-ary $(n+1, M, d+1)$-code exists.*

Proof: (i) (Only if part): Let $C \in (n, M, d)$-cod. We construct $C' \in (n+1, M, d')$-cod (some d') as follows.

For each $x \in C$ let $x' = x0$ if $w(x)$ even and $x' = x1$ if $w(x)$ odd.

Note that $d \leq d' \leq d+1$. But every x' has even weight by construction so d' is even by Lemma 0.24. Hence $d' = d+1$.
(ii) (If part): Let $D \in (n+1, M, d+1)$-cod$_2$. Take $x, y \in D$ such that $d(x, y) = d+1$. Find a digit, the k-th say, where they differ. Construct $D' \in (n, M, d')$-cod$_2$ by $D' = \pi_k(D)$. Note that $d \leq d' \leq d+1$. But $d(x', y') = d(x, y) - 1 = d$. Hence $D' \in (n, M, d)$-cod$_2$. $\square$

Corollary: If d odd then $A_2(n+1, d+1) = A_2(n, d)$.

0.27. Lemma. $A_q(n, d+1) \leq A_q(n, d)$.

Proof: Let C be an optimal $(n, M, d+1)$-code, so $M = A_q(n, d+1)$. Choose $x, y \in C$ with $d(x, y) = d+1$. Assume x, y differ in k-th digit. Remove x from C and replace it with x':

$$x' = \pi_k^{y_k}(x)$$

New code C' contains x' and y and $d(x', y) = d$ by construction, so $d(C') \leq d$. Let $z, w \in C'$. If neither is x' then $z, w \in C$ so $d(z, w) \geq d+1 > d$. If $z = x'$ (say) then

$$d+1 \leq d(x, w) \leq d(x, x') + d(x', w) = 1 + d(z, w)$$

so $d(z, w) \geq d$. Thus $d(C') \geq d$, so $C' \in (n, M, d)$-cod, so $A_q(n, d) \geq M$. $\square$

This gives us one last bound on $A_q(n, d)$:

Theorem 0.28. $A_q(n+1,d) \leq qA_q(n,d)$

Proof: Let C be an optimal q-ary $(n+1,M,d)$-code. Define $C_i = C \cap \pi_{n+1}^i(C)$. Clearly $C = \sqcup_{i \in S} C_i$ so $M = |C| = \sum_{i \in S} |C_i|$. Thus at least one of the C_is has order at least M/q. Choose such a C_i ($i = k$, say) and construct C' from it by deleting the last digit of each codeword:

$$C' = \pi_{n+1}(C_k)$$

Since $C_k \subset C$ we have $d(C_k) \geq d(C) = d$. But $d(C') = d(C_k)$ since all codewords in C_k agree in the last digit. Hence C' is a q-ary (n, M', d')-code with $M' \geq M/q$ and $d' \geq d$, so $A_q(n,d') \geq M/q$. But $d' \geq d$ so by (iterated use of) the Lemma above

$$A_q(n,d) \geq A_q(n,d') \geq M/q = A_q(n+1,d)/q$$

□

Examples and exercises

0.29. Example. Given $A_2(10, 3) \leq 79$ it follows that $A_2(11, 3) \leq 2 \times 79 = 158$.

0.30. Exercise. Use the above theorem to give an alternative proof of the singleton bound.

0.31. Exercise. For each of the following triples (n, M, d) construct, if possible, a binary (n, M, d) code:

$$(6, 2, 6) \qquad (3, 8, 1) \qquad (4, 8, 2) \qquad (8, 40, 3)$$

If no such code exists, prove it.

Answer:

A q-ary repetition code has $M = q$ and $d = n$ for any q, n. Our first case is an example of this: $\{000000, 111111\}$ is a (6,2,6) code.

As we have set things up, all codewords are necessarily distinct. This means that d is necessarily at least 1. To make a $d = 1$ code, then, all we have to do is make any code at all. The biggest q-ary length n code has $M = q^n$ (just include every possible codeword). For binary $n = 3$, therefore, this biggest code has $M = 8$.

That is, for (3,8,1):

$$\{000, 001, 010, 011, 100, 101, 110, 111\}$$

is the unique such code.

For our third case we can use the parity idea (proof of Theorem 0.26) to increase the distance by 1 from our (3,8,1) code:

$$\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$$

For our fourth case it is no longer obvious how to construct a code. Under the circumstances it is prudent to check if such a code is impossible, by checking the BP and singleton bounds. In this case one finds that the BP bound fails, so there is no such code.

Graphs and codes

An (undirected) *graph* G is a set V_G of vertices together with a set E_G of edges between them (for a more careful definition see the Chapter on Graphs online).

A *complete* graph is a graph in which every pair of vertices is connected by one edge.

A graph morphism $\phi : G \to G'$ is a map $\phi : V_G \to V_{G'}$ such that $(v_1, v_2) \in E_G$ implies $(\phi(v_1), \phi(v_2)) \in E_{G'}$.

0.32. Exercise. Consider the graph $G(n,k)$ each of whose vertices is a 2-ary sequence of length n; with an edge (x,y) whenever $d(x,y) \geq k$. A 2-ary length n code C is any subset of the vertex set of $G(n,k)$. If $G(n,k)$ restricts to the complete graph on C then $d(C) \geq k$.

(a) Prove it!

(b) Write down a maximal complete subgraph of each of the following: $G(3,3)$, $G(4,3)$, $G(5,3)$.

(c) If there is a complete graph of order l in $G(n,k)$ (l vertices) then there is a complete graph of order l including the vertex 000...0. Prove it.

(d) Let $\Psi : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ denote swapping the first two entries in the sequence (e.g. $\Psi(10111) = 01111$). Then Ψ defines a graph homomorphism from $G(n,k)$ to itself. Prove it. (Can we say more?)

ANSWERS: (a) Try this yourself. Note that it says that $A_2(n,k)$ is the size of a maximal complete subgraph in $G(n,k)$.
(b) We give our complete graph as a list of vertices in each case:
$G(3,3)$: $\{000, 111\}$ (equally good would be $\{001, 110\}$, but it will be clear than neither subgraph can be enlarged without losing the completeness property);
$G(4,3)$: $\{0000, 1110\}$;
$G(5,3)$: $\{00000, 11100, 10011, 01111\}$.
(c) If we change the first entry in every vertex sequence in $G(n,k)$ (from 0 to 1 or from 1 to 0) then the Hamming distances between vertices are not changed. The same applies if we change any given entry in every sequence simultaneously. In this way we may take any vertex (in a complete subgraph, say) and change it to 000...0 without changing the edges in the subgraph (so it remains as the complete graph). $\square$

(d) For every pair of vertices $d(x, y) = d(\Psi x, \Psi y)$, since the first two entries are interchanged in *both*. In fact Ψ gives a graph isomorphism of $G(n, k)$ with itself. But of course Ψ would not fix some arbitrary subset C in general.

Finite fields

We have repeatedly thought of S^n as if it were something like $\mathbb{R}^n$, that is, as if it were a vector space, and $C \subset S^n$ a vector subspace. Now we want to go further and think of strings

$$x = x_1 x_2 ... x_n = (x_1, x_2, ..., x_n)$$

as vectors, so that we can add them, and multiply by scalars.

In its simplest form this means that we want S itself to be like $\mathbb{R}$, in the sense of having addition and multiplication defined (perhaps even subtraction, and division by 'non-zero' elements). But S cannot *be* $\mathbb{R}$, since it is finite.

The composition requirements are summarised by saying that we want S to be a field. We should recall the definition of field; and then see if we can think of any *finite* fields that we could use for our alphabet.

The definition of field is quite long. We can break it up a little into stages:

0.33. Definition. A commutative ring is a set F equipped with 2 closed associative and commutative operations

$$+ : F \times F \to F, \qquad \times : F \times F \to F$$

(we will write ab for $\times(a, b) = a \times b$), such that:
(1) $\times$ is distributive over $+$:

$$a(b + c) = (ab) + (ac)$$

(2) there is an additive identity element $0 \in F$, so that

$$a + 0 = 0 + a = a \qquad \forall a$$

(3) there is a multiplicative identity element $1 \in F$, so that

$$a1 = 1a = a \qquad \forall a$$

(4) Every $a \in F$ has an additive inverse $-a$ such that
$a + (-a) = 0$.

0.34. Example. The integers form a commutative ring.

Fields

0.35. Definition. A field is a commutative ring such that
(5) Every $a \in F \setminus \{0\}$ has a multiplicative inverse a^{-1} such that $a(a^{-1}) = 1$.

0.36. Example. The obvious example is the real numbers. The rational numbers also work. As do the complex numbers.
The integers do not work, since 2 has no integer multiplicative inverse.

0.37. The challenge is to find finite sets F that can have all these properties. A great source of such examples comes from thinking about modular arithmetic:

Define a relation of congruence modulo 5 on $\mathbb{Z}$ by $a \cong b$ if $a - b = 5n$ for some integer n.

It is easy to see that this is an equivalence relation. The equivalence classes are:

$$[0] = ..., -10, -5, 0, 5, 10, ...$$

$$[1] = ..., -10 + 1, -5 + 1, 0 + 1, 5 + 1, 10 + 1, ...$$

and indeed for $r = 0, 1, 2, 3, 4$:

$$[r] = ..., -10 + r, -5 + r, 0 + r, 5 + r, 10 + r, ...$$

And miraculously ...

Miraculously, when we do ordinary integer arithmetic we find that it respects these classes. That is, if $a + b = c$ and a, b are congruent to a', b' respectively then $a' + b'$ is congruent to c. Example:

$$1 + 2 = 3 \qquad 21 + (-98) = -77$$

In this sense we can define arithmetic on the classes mod.p (where at this stage p is any natural number). The resultant structure of integer arithmetic mod.p is denoted $\mathbb{Z}_p$. Thus $\mathbb{Z}_p$ is a set with $+$ and $\times$ which are commutative and associative, distributive...

0.38. Exercise. Check this!

...with additive and multiplicative identity; and additive inverse. Example: For $p = 5$ the additive inverses of $[0], [1], ...$ are given by

$$[0] + [0] = [0] \qquad [1] + [4] = [0] \qquad [2] + [3] = [0]$$

so that [0]=-[0]; [4]=-[1] and [3]=-[2].

What about multiplicative inverses? Is there an $[x]$ such that $[2][x] = [1]$?
If we are working in $\mathbb{Z}_5$ then: Yes! $[2][3] = [6] = [1]$. And $[4][4] = [16] = [1]$.

Thus

Theorem 0.39. *$\mathbb{Z}_5$ is a field.*

On the other hand $\mathbb{Z}_4$ is a commutative ring, but not a field. The complete row of the multiplication table for $[2]$ is

$$[2][0] = [0] \qquad [2][1] = [2] \qquad [2][2] = [0] \qquad [2][3] = [6] = [2]$$

Since none of the right hand sides is $[1]$ we see that $[2]$ does not have a multiplicative inverse.

In fact

Theorem 0.40. *(i) $\mathbb{Z}_p$ is a field iff p is prime.*
(ii) there is a field of order q iff $q = p^e$ where p is prime and $e \in \mathbb{N}$.
(iii) two fields of the same order are isomorphic.
Part (i) can be proved as an exercise.
Part (ii) is standard in algebra textbooks, but for now we will content ourselves with understanding the statement.
Part (iii) just says that when we have understood part (ii) we will have a handle on *all* finite fields!

So, what about part (ii)? Part (i) tells us how to construct the fields of prime order; and that the fields of order p^2 and so on are *not* $\mathbb{Z}_{p^2}$ and so on.
...so what are they?

Fields and field extensions

One way to address this question is to think about how the rational field sits inside the real field; and the real field inside the complex field. We can ask ourselves what happens when we adjoin $i = \sqrt{-1}$ to $\mathbb{R}$ and try to make a field containing these objects. Since a field is closed under addition we see immediately that the smallest field containing $\mathbb{R}$ and i is $\mathbb{C}$. On the other hand if we adjoin i to $\mathbb{Q}$ we can construct a 'complex rational field' bigger than $\mathbb{Q}$ but smaller than $\mathbb{C}$.

One way of thinking of this is that we have added to $\mathbb{Q}$ a new number v, which number obeys $v^2 + 1 = 0$. We don't really need to know too much else about this number! We can already check the axioms:

A general element of the field can be written in the form $a + bv$ where $a, b \in \mathbb{Q}$. Adding obviously works:

$$(a_1 + b_1 v) + (a_2 + b_2 v) = (a_1 + a_2) + (b_1 + b_2)v$$

and multiplying (using $v^2 = -1$):

$$(a_1 + b_1 v)(a_2 + b_2 v) = (a_1 a_2) + (a_1 b_2 + a_2 b_1)v + (b_1 b_2)v^2$$

$$= ((a_1 a_2) - (b_1 b_2)) + (a_1 b_2 + a_2 b_1)v$$

The multiplicative inverse is given by $v^{-1} = -v$, since

$$v(-v) = -v^2 = 1$$

and more generally by:

Exercise!

0.41. Example. What happens if we further extend this field by adding in an object w obeying $w^2 - 2 = 0$?

0.42. The idea for finite fields is to make such extensions to the prime fields $\mathbb{Z}_p$ (p prime). Let us consider the prime $p = 2$, and try to extend the field $\mathbb{Z}_2$. We start by adding in an element that obeys a polynomial equation. We might as well start with a quadratic. Since we want to end up with 'coefficients' in $\mathbb{Z}_2$ the coefficients in the polynomial need to be in $\mathbb{Z}_2$. There is then only one irreducible polynomial available: $f(x) = 1 + x + x^2$. Adjoining a root of f to $\mathbb{Z}_2$ we get a number system consisting of $\{0, 1, x, 1 + x\}$, and that's it! The inverse of x is $1 + x$, since

$$x(1 + x) = x + x^2 = -1 = 1 \qquad (mod.p)$$

This field is called F_4.

More generally we can adjoin a root of an irreducible polynomial of degree e and get F_{2^e}. More generally still, F_{p^e}.

Note that the polynomial $f(x)$ we should use should have the property that it does not have a root in the original field (so the root is a 'new element'). In general we also require that there is no lower-order polynomial having the same root (i.e. $f(x)$ does not factorise). This is what we mean by 'irreducible polynomial'.

We said some time ago that coding is interested in the way the code C sits as a subset in the set of all possible received words (i.e. it is interested in the minimum distance $d(C)$ and so on). From this point of view, the precise choice of symbols used in codewords is not directly relevant. However, realistically, the message itself is quite likely to take the form of strings of letter from some human alphabet — and the recovery of the correct letters at the end of the process is the essential aim. In practice, then, since we are about to start using elements of finite fields to create codes, the question arises: How can we use finite fields to represent our familiar alphabet?

This is the same question as to ask how we can use *any* random set of symbols to represent our alphabet. Doing this is a vital step, if we are going to use new symbol sets. But it is, of itself, essentially trivial. Here we are not trying to maximise Hamming distance or anything like that, so any surjective map from the alphabet to some set of strings of symbols from the new symbol set will do. Thus if we have an alphabet with 26 letters in it (say!), we can represent it with some other symbol set, so long as there are at least 26 codewords available.

0.43. Example. The 26 letters of the alphabet $\{A, B, C, ..., Z\}$ may be represented in $\mathbb{Z}_3^3$ by $A \mapsto 001$, $B \mapsto 002$, $C \mapsto 010$, $D \mapsto 011$, $E \mapsto 012$, ..., $Z \mapsto 222$. This uses up 26 of the $3^3 = 27$ elements of $\mathbb{Z}_3^3$, so we may also represent 'space' by 000.

Linear codes

Our original idea was to be able to think of S^n as a set of vectors, by making S a field. The analogy was with the case $\mathbb{R}^n$, which is a set of n-component vectors forming a vector space.

If F is a field then F^n is an n-dimensional vector space over F. Addition is component-wise, as usual.

We say that code $C \subset F^n$ is a *linear* code if it is a linear subspace of F^n.

0.44. Example. Let
$V = \mathbb{Z}_2^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. Then
$C = \{000, 001, 010, 011\}$ is a subspace.
This is analogous to the fact that $\{(0, y, z) \mid y, z \in \mathbb{R}\}$ is a subspace of the *infinite* space $\mathbb{R}^3$. A basis of $\mathbb{R}^3$ is $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$, and a basis of the subspace is $\{(0, 0, 1), (0, 1, 0)\}$.

A basis of V is $\{(0,0,1),(0,1,0),(1,0,0)\} = \{001, 010, 100\}$, and a basis of C is $\{001, 010\}$.

Examples/Exercises

0.45. Example. Show that if $C, C' \subset F^n$ are linear codes then $C \cap C'$ and $C + C' := \{u + u' \mid u \in C,\ u' \in C'\}$ are also linear codes. When is the code $C \cup C'$ also linear?

0.46. Picking a code at random from $P(F^n)$, it is likely to be non-linear. However "most of the codes currently studied and used are linear" (Jones and Jones, 2000). We will now see why.

When $C \subset F^n$ is linear, and of dimension k as a vector space, then $M = |C| = |F|^k$. We call C a linear $[n, k]$-code.

0.47. The *rate* of a code is

$$R = R(C) = \frac{\log_q M}{n}$$

so for a linear code

$$R = k/n$$

Thus the bigger k is, the more information we transmit; the bigger n is, the longer it takes to transmit. But of course the bigger $n - k$ is the more checking we are doing, so the better we can confirm or protect the information.

Examples

Let us now examine some examples of linear codes. In particular, which of the codes we already looked at are linear?

If $S = F$ is a field then the repetition code $R_n \subset F^n$ is linear of dimension 1. Example: $11...1 + 11...1 = 22...2$.

Parity check codes

The *parity-check* code $P_n \subset F^n$ consists of all vectors u such that

$$\sum_i u_i = 0$$

We can consider the first $n-1$ digits as information, and u_n as a check digit, simply defined as

$$u_n = -\sum_{i=1}^{n-1} u_i.$$

Since it is defined by a linear equation this code is linear. It is a $[n, n-1]$-code, so $M = q^{n-1}$ and $R = n - 1/n$.

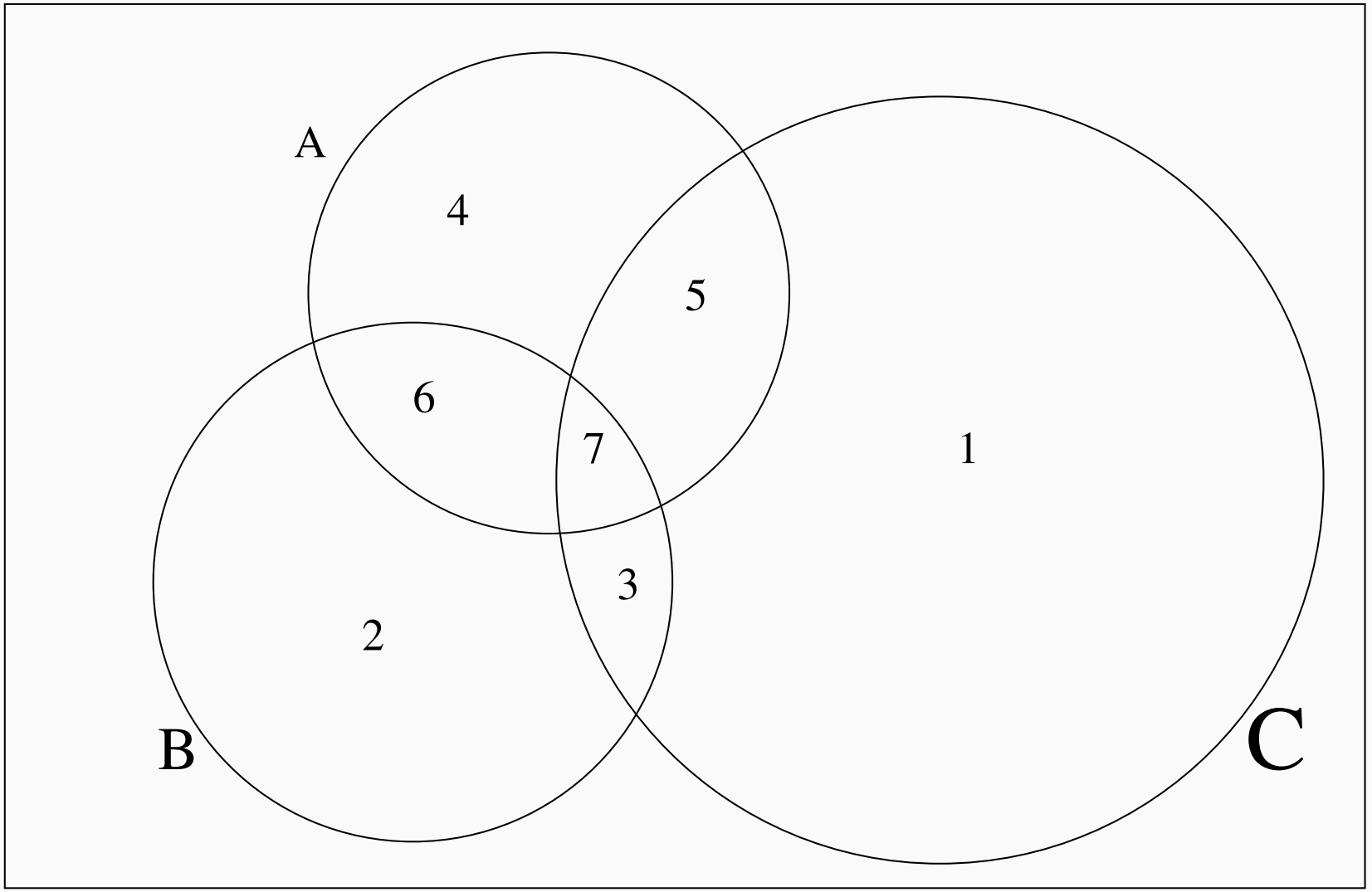


Figure 5:

0.48. Example. Consider the Venn diagram for sets A, B, C in Figure 5. Suppose we want to encode an element a of $\{0,1\}^4$ as a codeword $u \in S^7$. We will assign the 7 digits to the 7 regions in the figure as numbered. We set $u_3 = a_1$, $u_5 = a_2$, $u_6 = a_3$, and $u_7 = a_4$. We now want to set u_1, u_2, u_4 for collateral (checking) information. We set u_4 so that the sum of digits assigned in set A (i.e. u_4, u_5, u_6, u_7) is zero in binary. We set u_1, u_2 similarly considering C and B.

The code H_7 consists of all codewords $u \in F_2^7$ written in this way.

Since H_7 is determined by linear equations between variables u_i it is a linear code. There are 2^4 choices for a, and these fix u, so $M = 16$. Indeed H_7 has basis $v_1 = 1110000$, $v_2 = 1001100$, $v_3 = 0101010$, $v_4 = 1101001$. Thus the dimension is 4.

We will come back to this example later.

Linear algebra/linear combinations

Here are some quick reminders on linear algebra:

A linear combination of a set of vectors $V = \{v_i\}$ is a form like

$$v = \sum_i a_i v_i$$

Obviously we have

$$0 = \sum_i 0.v_i$$

(on the left we mean the zero vector; on the right the 'scalar/field element/number' 0).

The set of all vectors expressible as linear combinations of V is called the span of V.

Linear independence

0.49. Definition. A set of vectors is linearly independent if the *only* way to linearly combine them to get 0 is with all coefficients 0.

A linearly independent spanning set for a vector space is called a basis.

Basis Theorem

Theorem 0.50. *Let C be a non-trivial (i.e. non-zero) subspace of V, a vector space over F_q. Then*
(1) C has a basis.
Let $B = \{v_1, v_2, ..., v_k\}$ be a basis for C. Then
(2i) every vector in C can be uniquely expressed as a linear combination in B.
(2ii) $|C| = q^k$.
Proof: Exercise.

Note that any two bases for C have the same order, k. Call this number $\dim C$.

Examples

0.51. Example. (i) F_q^n has a basis $\{100..00, 010..00, ..., 000..01\}$ consisting of n vectors.
(ii) $C = \{000, 001, 010, 011\}$ is a subspace of $\mathbb{Z}_2^3$ with basis $\{001, 010\}$.
(iii) Is $C = \{000, 001, 002, 010, 020, 011, 022\}$ a subspace of $\mathbb{Z}_3^3$?
No! The dim is not a power of 3.

0.52. Proposition. *Let F be a finite field of characteristic p. Then F is itself a vector space over $\mathbb{Z}_p$.*

Proof: Exercise.

Notation

0.53. Definition. A q-ary $[n, k, d]$-code is a linear code in F_q^n of dim k and minimum distance d. Write $[n, k, d] - cod$ for the set of all such (with q understood).

Thus $C \in [n, k, d] - cod$ implies $C \in (n, q^k, d) - cod$, but the converse is false.

0.54. Example. Our first three examples are all binary linear codes: $C_1 \in [2, 2, 1] - cod$; $C_2 \in [3, 2, 2] - cod$; $C_3 \in [5, 2, 3] - cod$. Exercise: check this.

Minimum weight/distance

Recall that for a general code we need $\frac{1}{2}|C|(|C|-1)$ calculations to compute $d(C)$. We can radically reduce this for a linear code.

To see this first note that

$$d(x,y) = w(x-y)$$

Thus

Theorem 0.55. *For a linear code let*

$$w(C) = min\{w(x) \mid x \in C \setminus \{0\}\}$$

(here we write 0 for the appropriate 000..0 sequence, for convenience). Then

$$w(C) = d(C).$$

Proof: Exercise.

Specifying a linear code

For linear codes we usually just give a basis rather than listing out the whole thing.

Generator matrix

0.56. Definition. A $k \times n$ matrix is called a generator matrix for C if its rows form a basis for C.

0.57. Example. C_3 has generator matrix

$$G = \begin{pmatrix} 01101 \\ 10110 \end{pmatrix}$$

However in computing $d(C)$ it is NOT enough to find the minimum weight among the basis vectors! For example

$$G = \begin{pmatrix} 1111 \\ 1110 \end{pmatrix}$$

has min weight 3, but $d(C) = 1$.

.

Theorem 0.58. *Let G generate C. Any matrix obtained from G by*
(R1) permuting rows
(R2) multiplying a row by a non-zero scalar
(R3) adding one row to another
generates the same code.

Example

0.59. Example. Show that the 3-ary linear codes generated by

$$G = \begin{pmatrix} 210222 \\ 012101 \\ 011112 \end{pmatrix}$$

and

$$G' = \begin{pmatrix} 100201 \\ 010120 \\ 001022 \end{pmatrix}$$

generate the same code. Deduce $d(C)$.

Clues: start by subtracting row two from row three in G. Then subtract row 2 from row 1. Then row 3 from row 2. Then multiply row 1 by the scalar 2. How does it look now?!

Obviously $d(C) \leq 3$, since there is a weight 3 row in G'. But to see if this bound is saturated (it is) you still have some work to do!

Equivalent codes

0.60. Definition. Codes C, C' are equivalent (write $C \sim C'$) if there is a one-to-one mapping

$$\phi : C \to C'$$

such that

$$d(x, y) = d(\phi(x), \phi(y))$$

for all x, y. In particular $d(C) = d(C')$.

0.61. Exercise. Check $C \sim C'$ is an equivalence relation, i.e. a reflexive, symmetric, transitive relation.

Theorem 0.62. *Let C be a linear code generated by G. Let G' be obtained from G by*
(C1) permuting columns
(C2) multiplying a column by a non-zero scalar $a \in F_q$.
Then G' generates C' an equivalent linear code to C.
Proof: Exercise (optional!).

By using all the row and column operations you can always reduce G to a standard form

$$G' = \left(\begin{array}{c|c} 100..0 & A_{11}A_{12}..A_{1,n-k} \\ 010..0 & A_{21}A_{22}..A_{2,n-k} \\ \ldots & \\ 000..1 & A_{k1}A_{k2}..A_{k,n-k} \end{array}\right) = [1_k|A]$$

where 1_k is the $k \times k$ unit matrix and A has entries in F_q.

0.63. Example. A binary [5,3,d]-code is generated by

$$G = \begin{pmatrix} 11111 \\ 10011 \\ 11001 \end{pmatrix} \sim \begin{pmatrix} 11111 \\ 01100 \\ 00110 \end{pmatrix} \sim \begin{pmatrix} 10011 \\ 01010 \\ 00110 \end{pmatrix}$$

0.64. Exercise. Let C_i be the 3-ary code generated by G_i, where

$$G_1 = \begin{pmatrix} 1011 \\ 0112 \end{pmatrix}, \qquad G_2 = \begin{pmatrix} 1011 \\ 0111 \end{pmatrix}$$

For each of $i = 1, 2$, list C_i and hence compute $d(C_i)$. Is C_i perfect?

Exercise Answer hints

C_1: Let's call the two row vectors v_1 and v_2, then we can write out all linear combinations and hence all elements systematically:

$av_1 + bv_2$	$a = 0$	1	2
$b = 0$	0000	1011	2022
1	0112	1120	2101
2	0221	1202	2210

Thus $d(C_1) = w(C_1) = min(w(x \in C_1^*)) = 3$.

Now consider the size of a 1-ball around a codeword x in this case. (1-ball, since $2t + 1 = 3$ gives $t = 1$.) It includes the codeword itself, plus the 8 words differing from x in a single position (4 positions; $q - 1 = 2$ ways to differ in each position). The total 'space' occupied by $M = 9$ such balls is thus 9×9. So... you now just have to compare with the total size of the code 'universe' here...

Hints Contd

C_2: We can proceed systematically in the same way as above.

Alternatively,... Consider $v_1 = 1011$ and $v_2 = 0111$ here. And consider some specific linear combinations $av_1 + bv_2$. For example $v_1 + 2v_2 = 1011 + 0222 = 1200$. Thus $w(C)$ is at most 2.

On the other hand both vectors v_1, v_2 have sum of entries (3-ary 'parity') zero. So all combinations also have sum zero. Thus the only way to get a codeword with form 00*0, say, $(w(x) \leq 1)$ is with $* = 0$. Thus $w(C) = 2$. Thus $d(C) = 2$.

... Thus C_2 not perfect (since t is smaller than for C_1, which has the same M and 'universe') (or on general grounds, since d even).

Encoding

.

Given C, a linear code over F_q (i.e. a subset of F_q^n for some n) generated by G, we have a natural identification between C and F_q^k ($k = dim\ C$, *not* the same as n, the length of the code). Each $x \in C$ is uniquely expressible as

$$x = \sum_{i=1}^{k} a_i v_i$$

(the v_is are the rows of G in the natural order). So

$$x \leftrightarrow (a_1, a_2, ..., a_k) \in F_q^k$$

is a one-to-one correspondence.

We think of the $a = (a_1, ..., a_k)$ vectors as the *message words* of the code, and the n-tuples x as the codewords representing them.

Note that the encoding map

$$a \to x$$

is then simply

$$x = aG$$

That is, right multiplication by the generating matrix — a linear map!

0.65. Example. Let C be 3-ary and generated by

$$G = \begin{pmatrix} 10010 \\ 01010 \\ 00102 \end{pmatrix}$$

Encode the messagewords 000, 101 and 122.

Clearly $000 \to 00000$, so we need

$$101 \to (101)G = (10112)$$

$$122 \to (122)G = (12201)$$

Note that the first three digits of the codeword are the same as the messageword. This always happens if G is in the standard form. The other digits are then 'check' digits.

This makes the last part of decoding trivial:

$$\textit{messageword} \overset{\textit{encode}}{\to} \textit{codeword} \overset{\textit{transmit}(\textit{noise})}{\to} \textit{received vector}$$

$$\overset{\textit{project}}{\to} \textit{nearest codeword} \overset{\textit{interpret}}{\to} \textit{decoded messageword}$$

The last step is just to drop off the check digits.

Coset decoding

Error vector

Our picture above raises a key point. When $x \in F_q^n$ is transmitted down a noisy channel y is received. Define the error vector

$$e = y - x$$

Then the number of transmission errors is $w(e)$. (Of course no one knows both x and y for sure...)
We want an algorithm which decides from y which x was (probably) sent; or equivalently, what e has occurred.

0.66. Definition. Suppose $C \in [n, k, d] - cod_q$ (some d) and $a \in F_q^n$. Then set

$$a + C = \{a + x \mid x \in C\}$$

is called a coset of C in F_q^n.

.

Theorem 0.67. *[Lagrange] (a) The cosets of a linear code $C \subset F_q^n$ partition F_q^n.*
(b) Each coset has size q^k.

Proof: (Idea) Think of C as a subspace (such as a plane in $\mathbb{R}^3$ through the origin). We can think of a as shifting this subspace parallel-ly away from the plane; in other words to a new plane not including the origin.
We dont have $\mathbb{R}^3$, but the same idea works. $\square$

Example

0.68. Example. Let C be 2-ary generated by

$$G = \begin{pmatrix} 1011 \\ 0101 \end{pmatrix}$$

That is $C = \{0000, 1011, 0101, 1110\}$. Cosets:

$$0000 + C = C$$

$$1000 + C = \{1000, 0011, 1101, 0110\} = 0011 + C \qquad (etc)$$

and so on.

Coset leaders

Given any subset U of F_q^n (such as a coset), we may partition U into subsets containing words of equal weight. Among these will be a subset of words in U of *least* weight. (For example, if we consider the whole of F_q^n then there is always a fixed-weight subset containing just the word of weight zero.)

Henceforth we assume that we always have a way of choosing a single word from any such subset. (If we have totally ordered the words in F_q^n then we could simply take the first one in the order induced on the subset, say.)

Now suppose that the subset we have in mind is a coset. The chosen vector of min weight in a coset is called the *coset leader* (for that choice). E.g. in $\{0100, 1111, 0001, 1010\}$ either 0100 or 0001 could be chosen as leader.

Standard array

(**10.1**) We can use the idea of coset leaders to generate an arrangement of F_q^n called a *standard array* for C. Assuming as before that we have a way to choose a word from a set (via a total order, say), then we can do this algorithmically:

(i) make a row list of the codewords of C, with 00..0 on the left. This row is coset $00..0 + C$, with 00..0 as coset leader; arranged in some chosen order.

(ii) choose any distinct vector a_1 of min weight in $F_q^n \setminus C$ and row list $a_1 + C$ in the obvious order (i.e. with $a_1 + c$ under codeword c). This has a_1 as coset leader.

(iii) choose any a_2 not already listed, of min weight, and row list $a_2 + C$.

(iv) repeat until all words of F_q^n appear.

Note that there were two kinds of choices in the construction of the standard array: (1)the order in which to write out the row C after 00...0; (2) the choices of coset leaders (the column below 00...0). As we shall see, the first choice has no real bearing on decoding in what follows. The second choice can affect decoding (but all such choices are equally 'good' in probablistic terms).

In our example 0.68 the standard array (for an obvious set of choices) is

$$\left(\begin{array}{c|c|c|c} 0000 & 1011 & 0101 & 1110 \\ 1000 & 0011 & 1101 & 0110 \\ 0100 & 1111 & 0001 & 1010 \\ 0010 & 1001 & 0111 & 1100 \end{array}\right)$$

(don't worry — we won't always need to write out this whole table — see section 13).

(**10.2**) We are now ready to explain coset decoding.

Note that a given standard array A determines, for each word y in F_q^n, a coset leader $e_A(y)$ (the first word in the row of y); and a codeword $c_A(y)$ (the first word in the column of y). For example, the coset leader associated to 1010 in the array above is 0100. Thus if we receive y, we may associate two other words to it, related by

$$e_A(y) = y - c_A(y)$$

Coset decoding:
if we receive y, we decode it as the codeword $c_A(y)$ appearing in the column containing y.

.

IS this a good strategy?
In coset decoding we are effectively assuming that the actual error from the transmitted codeword x

$$e = y - x$$

is the coset leader $e_A(y)$ of coset $y + C$.

Suppose the actual error e *is* a coset leader. Then $y = x + e$, so y does lie in the coset with leader e, so $e_A(y) = e$ and $c_A(y) = x$. That is, our decoding is correct.

On the other hand, if the actual error is not a coset leader, then by assuming that it is, we are bound to get the decoding wrong.
By choosing coset leaders to have min weight, we always decode y as the (Hamming) nearest codeword to y (or at least one of the joint nearest). E.g. $y = 0110$ decodes as $x = 1110$ in our example.
That is, we assume the fewest errors possible.
In case of low single-digit error probability it is hopefully already clear that this is a good assumption — probablistically. (But see section 11 for details.)

Returning to our example code, note that $d(C) = w(C) = 2$. Thus it is not even single error correcting, so even single errors might not be corrected properly. (In fact a single error will be corrected if it occurs in the 1st, 2nd or 3rd digit, but not the 4th.) Specific instances:

messageword		*codeword*	*noisy channel*	*decode*	*truncate*	
	$\rightarrow$					
01		0101	0111 (*say*)	0101	01	(*correct*)
10		1011	1010	1110	11	(*incorrect*)

This glitch is precisely to do with the fact that we had a choice of coset leaders in $0100 + C$. We could have chosen 0001 instead, in which case the 4th digit errors would be recovered and the 2nd digit errors not recovered.

Probability of error correction/detection

As noted before, it is the probabilities of a successful outcome which really dictate the success of our coding methodology. We have accumulated a lot of technology since our last probability calculation, so now it is time to put it all together.

Suppose we transmit a linear code down a symmetric channel with symbol error probability p, then use coset decoding. Then we get the decoding of any received word y right if and only if our error correction is right. This happens in coset decoding if and only if the actual error $e = y - x$ is a coset leader. Thus for any transmitted codeword x

$$P_{corr}(x) = Prob(\text{error } e = \text{ one of the coset leaders })$$

(Note that this is independent of x!) In our example 0.68, therefore

$$P_{corr}(x) = P(e = 0000) + P(e = 1000) + P(e = 0100) + P(e = 0010)$$
$$= (1-p)^4 + 3p(1-p)^3$$

$$P_{err}(x) = 1 - P_{corr}(x)$$

Call this $P_{err}(C)$ since it depends only on C, not on x. It is the *word error rate* of the code.

More generally: Let C be any linear code whose coset leaders are $a_0 = 00..0$, $a_1, a_2, ..., a_l$. We have

$$P_{corr}(C) = \sum_{r=0}^{l} P(e = a_r) = \sum_{r=0}^{l} p^{w(a_r)}(1-p)^{n-w(a_r)} = \sum_{s=0}^{n} \gamma_s p^s (1-p)^{n-s}$$

where γ_s is the number of coset leaders of weight s.

.

How can we compute the γ_ss? In general it is hard, but:

Theorem 0.69. *If $d(C) \geq 2t+1$ then every vector of weight $\leq t$ is a coset leader for C. Hence*

$$\gamma_s = \binom{n}{s}(q-1)^s = |S_s(00..0)|$$

for $0 \leq s \leq t$. (Recall $S_s(00..0)$ is the sphere around 00..0.)

Proof

Proof: Consider the vectors in $B_t(00..0)$. Every vector lies in some coset, so if $y \in B_t(00..0)$ is *not* a coset leader then there exists z with $w(z) \leq w(y)$ and $x \in C$ $(x \neq 0)$ such that

$$y = x + z$$

But then

$$d(C) \leq w(x) = w(y - z) = d(y, z) \leq d(y, 0) + d(0, z)$$

$$= w(y) + w(z) \leq 2w(y) \leq 2t$$

This contradicts the first hypothesis, so y is a coset leader. $\square$

Theorem 0.70. *If C is a perfect $[n, k, 2t + 1]$-code then its coset leaders are precisely the vectors of weight $\leq t$.*

0.71. Example. Let C be a 3-ary $[11, 6, 5]$-code. What is $P_{err}(C)$?

By Theorem 0.69, $d = 5$ implies all vectors of weight ≤ 2 are coset leaders. Therefore

$$\gamma_0 = 1 \qquad \gamma_1 = \binom{11}{1} 2^1 = 22 \qquad \gamma_2 = \binom{11}{2} 2^2 = 220$$

(For $w > 2$, what is γ_w? We don't know, but let's press on!)

Therefore

$$P_{corr}(C) = \sum_{w=0}^{n} \gamma_w p^w (1-p)^{n-w} \geq \sum_{w=0}^{2} \gamma_w p^w (1-p)^{n-w}$$

$$= (1-p)^{11} + 22p(1-p)^{10} + 220p^2(1-p)^9$$

so

$$P_{err}(C) = 1 - P_{corr}(C) \leq 1 - ((1-p)^{11} + 22p(1-p)^{10} + 220p^2(1-p)^9)$$

In fact the bound is saturated, because this code is perfect. We know the weights of 1+22+220=243 of the coset leaders. But the number of cosets is

$$\frac{|F_q^n|}{|C|} = q^n/q^k = 3^{11}/3^6 = 3^5 = 243$$

so there *are* no more cosets!

If we use C for error detection, rather than error correction, the analogue of $P_{err}(C)$ is $P_{undetec}(C)$, that is, the probability that a word is received with undetected errors.

Again we transmit $x \in C$ and receive $y \in F_q^n$. The received vector has undetected errors iff $y \neq x$ but $y \in C$. That is, iff $e \in C \setminus \{00..0\}$.

The probability of this is again independent of x:

$$P_{undetec}(C) = \sum_{w=1}^{n} \delta_w p^w (1-p)^{n-w}$$

where δ_w is the number of codewords of weight w.

0.72. Example. $C = \{0000, 1011, 0101, 1110\}$. $\delta_1 = 0$, $\delta_2 = 1$, $\delta_3 = 2$, $\delta_w = 0$ $(w \geq 4)$. Thus

$$\begin{aligned} P_{undetec}(C) &= 0.p(1-p)3 + 1.p^2(1-p)^2 + 2.p^3(1-p) \\ &= p^2(1-p)(1-p+2p) = p^2(1-p^2) \\ &= 0.00009999 \qquad \text{if } p = 0.01 \end{aligned}$$

If y is received and $y \notin C$ we detect an error and request retransmission. How likely is this?

$$P_{retrans} = 1 - P(\textit{no error detected})$$

$$= 1 - (P(\textit{no errors}) + P(\text{error occurs but is not detected}))$$

$$= 1 - (1-p)^n - P_{\textit{undetec}}(C)$$

In our example

$$P_{\textit{retrans}}(C) = 0.039394 \qquad \text{if } p = 0.01$$

which is about 4%.

Dual codes

Notation: Here we use G^t (or G^T) to denote the matrix transpose. Recall the inner (or scalar) product $u.v$ of two vectors.

0.73. Example. In $\mathbb{Z}_2^4$: 1001.1101=1+0+0+1=0.

0.74. Definition. Given $C \subset F_q^n$, its dual code is

$$C^\perp = \{u \in F_q^n \mid u.v = 0 \; \forall \; v \in C\}$$

0.75. Lemma. *If C generated by G then $v \in C^\perp \iff vG^t = 0$.*

If U, V are vector spaces over F_q and

$$L : U \to V$$

is a linear map, then the range $L(U) \subset V$ is a subspace of V. Its dimension is called the rank of L.

Rank-Nullity

The set of vectors

$$ker\ L = \{u \in U \mid L(u) = 0\} \subset U$$

is a subspace of U, called the kernel of U. The dimension of the kernel is called the nullity of L. We have

$$rank\ L + dim(ker\ L) = dim\ U$$

(The Rank-Nullity Theorem from linear algebra.)

Let F be a field (such as F_q). Let us explicitly regard vector space $V = F^n$ as the space of n-component *row* vectors (as has been our convention throughout), i.e. as $1 \times n$ matrices. There is, formally, another realisation as *column* vectors — and even given the choice of row vectors, the explicit matrix representation of individual vectors $v \in V$ depends in principle on a choice of basis. But as soon as we fix all these choices, then ...

...Each $n \times k$ matrix H with entries in F defines a map

$$L_H : F^n \to F^k$$

by

$$v \mapsto vH$$

Equivalently each $k \times n$ matrix G with entries in F defines a map L^G by $v \to vG^T$.
Let $\alpha, \beta \in F$. Since $H(\alpha v + \beta w) = \alpha Hv + \beta Hw$ we see that L_H is a linear map.
If we consider the standard ordered basis for F^n then its image under L_H will be the set of row vectors in H. Thus

0.76. Lemma. *The dimension of $L_H(F^n)$ (the rank of L_H) is the same as the rank of H* as a matrix*.*
The same argument holds for L^G and the rank of G.[8]

[8] One can reconstitute all of this for the case where one regards vectors as *column* vectors, simply by 'transposing everything':
$v^T \mapsto (vG^T)^T = (G^T)^T v^T = Gv^T$.

Example

0.77. Example. Let's try a matrix with rank 1:

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} x+y & x+y \end{pmatrix}$$

Clearly the image space has dim=1 (albeit embedded in a 2d space).

Theorem 0.78. *If C an $[n, k]$-code over F_q (i.e. an $[n, k, d]$-code for some d), then $C^\perp$ is an $[n, n-k]$-code over F_q.*

Proof: Let G be a generator matrix for C, and consider the map

$$L : F_q^n \to F_q^k$$

defined by

$$L : v \mapsto vG^T$$

(note that G has k rows and n columns, so G^T has n rows and k columns). Then $C^\perp = ker\ L$ by Lemma 0.75. Thus $C^\perp$ is linear. Now

$$dim\ C^\perp = \ dim(ker\ L) = dim\ F_q^n - rank\ L \ = \ n - rank\ L$$

But $rank\ L \ = \ rank(G) \ = k$, since a generator matrix has full rank by definition.

□

0.79. Example. $C = \{000, 110, 011, 101\}$ over $\mathbb{Z}_2$ has dimension 2, so the dimension of $C^\perp$ is 3-2=1. We have

$$G = \begin{pmatrix} 110 \\ 011 \end{pmatrix}$$

and $v \in C^\perp$ iff

$$(v_1, v_2, v_3)G^t = (v_1 + v_2, v_2 + v_3) = (0, 0)$$

Over $\mathbb{Z}_2$ this holds iff $v_1 = v_2 = v_3$. Thus $C^\perp = \{000, 111\}$.

Theorem 0.80. *For all linear codes* $(C^\perp)^\perp = C$.

0.81. Definition. Any generator matrix H for $C^{\perp}$ is called a Parity Check Matrix (PCM) for C.

Theorem 0.80 says $x \in C$ iff $x \in (C^{\perp})^{\perp}$ iff $xH^t = 0$ (via Lemma 0.75). This says that we can think of C as the kernel of the linear map from F_q^n to F_q^{n-k} given by

$$x \mapsto xH^t$$

This says that the $n-k$ rows of H give the coefficients in $n-k$ linear equations which x must satisfy to be a codeword:

$$H_{11}x_1 + H_{12}x_2 + \ldots + H_{1n}x_n = 0$$

and so on. These are parity check equations (hence PCM).

0.82. Definition. The redundancy of a linear code is $r = n - k$, the number of extra digits added compared to the messageword.

Usually $r < k$ (fewer check digits than message digits), so H is smaller than G and the PCM is a more efficient way to define C than G is.
Given G, can we write down H?...

Theorem 0.83. *Let C be a $[n, k]$-code over F_q generated by*

$$G = [1_k | A]$$

where A is a $k \times (n-k)$ matrix (i.e. G is in standard form). Then $H = [-A^t | 1_{n-k}]$ is PCM for C.
Proof: Exercise.

Example

0.84. Example. 3-ary [6,4]-code generated by

$$G = \left(\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 & 2 \end{array}\right)$$

has PCM

$$H = \left(\begin{array}{cccc|cc} -1 & 0 & -2 & -2 & 1 & 0 \\ -1 & -2 & -1 & -2 & 0 & 1 \end{array}\right) = \left(\begin{array}{cccccc} 2 & 0 & 1 & 1 & 1 & 0 \\ 2 & 1 & 2 & 1 & 0 & 1 \end{array}\right)$$

.

0.85. Definition. A PCM H is in standard form if $H = [B|1_{n-k}]$ where B is a $(n-k) \times k$ matrix.
(Every linear code is equivalent to one whose PCM is in standard form.)

Syndrome decoding

We can use the PCM idea to make decoding more efficient. The idea is, if we receive a vector $y \in F_q^n$ we can compute which coset of C it lies in by computing its syndrome:

0.86. Definition. Let H be a PCM for a $[n, k]$-code C over F_q^n. The syndrome map of C is

$$S : F_q^n \to F_q^{n-k}$$

$$S(y) = yH^t$$

$S(y)$ is the syndrome vector of y. (Note this is a linear map.)

Note that $C = ker\ S$. In fact cosets of C are in 1-to-1 correspondence with syndromes.

0.87. Lemma. *Vectors $u, v \in F_q^n$ are in the same coset of C iff $S(u) = S(v)$.*

Indeed the number of cosets is q^{n-k}, which is the number of vectors in F_q^{n-k}, so cosets and syndromes are in bijective correspondence.

0.88. Example. (NB this is Example 0.68 revisited.)
Binary code generated by

$$G = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{array}\right)$$

gives PCM

$$H = \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{array}\right)$$

The coset leaders for C are 0000, 1000, 0100, 0010, and the syndromes: $S(0000) = 00$,

$$S(1000) = (1,0,0,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 11$$

$$S(0100) = (0,1,0,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 01$$

$$S(0010) = (0,0,1,0)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 10$$

If we receive $y = 1010$ then

$$S(y) = (1010)\begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = 01$$

so y is in the coset $0100 + C$. Thus ...

...we decode as $x = y - 0100 = 1110$.
Note that we no longer need most of the standard array; just the coset leaders and their syndromes: a syndrome look-up table.

Therefore we have a new decoding scheme:
(i) receive $y \in F_q^n$, calculate $S(y) = z \in F_q^{n-k}$.
(ii) look up z in table, i.e. find the coset leader l (say) such that $S(l) = S(y) = z$.
(iii) decode y as $x = y - l$.
This is much more efficient for large codes.

So, how do we compute $d(C)$ in all this?

Theorem 0.89. *Let C be a [n,k]-code over F_q with PCM H. Then $d(C) = d$ iff every set of $d-1$ columns of H is linearly independent, but there exists some set of d columns which is linearly dependent.*

Proof: Let c_i be the i-th column of H. If $x \in C$ has weight w then H has a set of w columns which is linearly dependent: x has w non-zero digits, $x_{i_1}, x_{i_2}, ..., x_{i_w}$ (say), and $xH^t = 0$ so

$$(0, .., x_{i_1}, .., 0, .., x_{i_2}, ..., x_{i_w}, .., 0) \begin{pmatrix} c_1^t \\ c_2^t \\ \vdots \\ c_n^t \end{pmatrix} = \sum_i x_i c_i = 0$$

so the set of w columns $\{c_{i_1}, c_{i_2}, ..., c_{i_w}\}$ is linearly dependent.

Conversely, to each LD set of columns one has a codeword x. If $d(C) = d$ then C has a codeword of weight $w = d$, but no codeword of weight $w = d - 1$. $\square$

Special cases:

$d(C) \geq 2$ iff no set of 1 columns is LD $\iff$ H has no zero columns.

$d(C) \geq 3$ iff no set of 2 columns is LD $\iff$ H has no parallel columns.

0.90. Example. What is $d(C)$ for the binary codes generated by

$$G_1 = \begin{pmatrix} 1011 \\ 0101 \end{pmatrix} \qquad G_2 = \begin{pmatrix} 10110 \\ 01101 \end{pmatrix}$$

giving

$$H_1 = \begin{pmatrix} 1010 \\ 1101 \end{pmatrix} \qquad H_2 = \begin{pmatrix} 11100 \\ 10010 \\ 01001 \end{pmatrix}$$

H_1 has no zero column, but has parallel, so $d(C_1) = 2$; while H_2 has no zero or parallel columns, so $d(C_2) \geq 3$. On the other hand $c_1 + c_3 + c_4 = 0$ for H_2, so $d(C_2) \leq 3$ (since $10110 \in C$). Thus $d(C_2) = 3$.

0.91. Example. Consider linear code C over $\mathbb{Z}_{11}$ with PCM

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & X \end{pmatrix}$$

This C has length 10 (number of columns of H), redundancy 2 (number of rows), so dimension 8. There are no parallel columns, so $d(C) \geq 3$. We have $c_1 - 2c_2 + c_3 = 0$ so $1910000000 \in C$, so $d(C) \leq 3$. Hence $d(C) = 3$ — it is a single error correcting code.

This code has a neat partial decoding scheme: ...

Since $d(C) = 3$ every vector of weight ≤ 1 is a coset leader of C (by our earlier result). There are 100 weight 1 vectors in $\mathbb{Z}_{11}^{10}$, namely all non-zero multiples De_i of all e_i (standard ordered basis elements). [9]

The syndrome of the coset led by De_i is given by:

$$S(De_i) = De_i H^t = (0, 0, .., 0, D, 0, .., 0) \begin{pmatrix} 11 \\ 12 \\ \vdots \\ 1\ i \\ \vdots \\ 1X \end{pmatrix} = (D, Di)$$

So from (D, Di) we get the coset leader: De_i.

[9] There are $11^{10}/11^8 = 121$ cosets altogether.

The partial decoding scheme is:
(i) receive $y \in \mathbb{Z}_{11}^{10}$, compute $S(y) = (A, B) \in \mathbb{Z}_{11}^{2}$.

$$(A, B) = (y_1, y_2, .., y_{10}) \begin{pmatrix} 11 \\ 12 \\ \vdots \\ 1\ i \\ \vdots \\ 1X \end{pmatrix} = (\sum_{i=1}^{10} y_i,\ \sum_{i=1}^{10} i y_i)$$

(ii) if $(A, B) = (0, 0)$ then $y \in C$: decode as $x = y$.
(iii) if A, B both nonzero assume single error occurred since (A, B) is $S(De_i)$ for some D, i. Decode as $x = y - Ae_i$ where $i = A^{-1}B$.
(iv) If only one of A, B is non-zero y is not in a coset led by a weight 1 or 0 vector.
Therefore at least 2 errors have occured. Request retransmission. ...

(This is why it is a *partial* scheme. We could have searched through the standard array for weight 2 coset leaders, but they will not be unique, so our 'best' guess will probably have some arbitrariness. Instead just get a retransmission.)

0.92. Example. decode $y = 1025234260 \in \mathbb{Z}_{11}^{10}$:

$$A = \sum_i y_i = 1+2+5+2+3 \ + \ 4+2+6 = 2+1 = 3$$

$$B = \sum_i i y_i = 1\times1+2\times0+3\times2+4\times5+5\times2+6\times3+7\times4+8\times2+9\times6 = 10$$

We are in case (iii) so assume error is $A = 3$ in digit
$i = A^{-1}B = 3^{-1} \times 10 = 4 \times 10 = 7$. Thus subtract 3 from y_7:
$x = 1025231260$.
(Exercise: check this is in C!)
On the other hand $y = 2610197034$ has $A = 0$ and $B \neq 0$ (check it!), so seek retransmission in this case.

Remark

0.93. Remark. This partial decoding generalises to $d = 2t + 1$ — all vectors of weight $\leq t$ coset leaders: list their syndromes. If receive y with $S(y)$ in the list decode it; else seek retransmission.

More Exercises

Encoding

Consider linear code C, with generator matrix G. The code is a certain copy of $F_q^k \hookrightarrow F_q^n$ (ideally chosen so that points are Hamming well separated in F_q^n).

So far, we took no account of frequency of use of messagewords, or any other differentiation among messagewords. Thus all points of C, as encodings of messagewords, are of equal standing. In particular there is no reason to try to make some further apart than others. Thus also there is no particular merit in one embedding of the set of messagewords in C over another.

We encode by

$$w \mapsto wG \in C$$

but there are many Gs corresponding to C. Thus for a given message, while fixing C we still get many different encodings.

If $G = G_s$ is in the standard form, we could call the resultant encoding the standard encoding.
If G is a row perm of G_s we might call this semistandard — the encoding of a message is already different. (This practical change should not be forgotten — noting that the 'code' as we define it has not changed; the PCM does not need to change; and the probablistic effectiveness of the code is not affected.) (An example follows shortly.)
If it is a row and column perm the code changes, and the PCM changes (albeit not in a deep way — the encoding is just permuted by the row perm).
Now read on.

1. The 26 letters of the alphabet may be represented in $\mathbb{Z}_3^3$ by $A \mapsto 001$, $B \mapsto 002$, $C \mapsto 010$, ..., $Z \mapsto 222$. Let us also represent 'space' by 000.

We are given the parity check matrix

$$H = \begin{pmatrix} 101201 \\ 011100 \\ 000011 \end{pmatrix}$$

of a linear code C. That is, $w \in C$ iff $Hw^t = 0$. (As usual we write simply 0 for the zero vector, where no ambiguity can arise.)

For example $H(100012)^t = 0$, so $100012 \in C$.

1.1 Note that H is not in standard form. Confirm that

$$G = \begin{pmatrix} 221000 \\ 120100 \\ 200021 \end{pmatrix}$$

is a generator matrix for C.

.

ANSWER: This means we have to check that the rows of G are a basis for C. We check (I) that the rows are linearly independent — so that they are a basis for *something*. We confirm this, for example, by noting how the rows differ in columns 3,4 and 6.
(II) that $GH^t = 0$ (by an explicit calculation) — this checks that the rows all *belong* to C.
(III) that the rows span C. Since the dual code has dimension 3 (the number of rows of H) we know that C itself has dimension $6 - 3 = 3$, so G must have 3 rows.

1.2 Write down another generator matrix for C.

Compute the encoding of the letter E, both by G and by your own choice of alternative generator matrix.

ANSWER: For example

$$G' = \begin{pmatrix} 120100 \\ 221000 \\ 200021 \end{pmatrix}$$

The encoding of E is different by G and by G'. We have

$$(012)G = (012)\begin{pmatrix} 221000 \\ 120100 \\ 200021 \end{pmatrix} = 220112$$

$$(012)G' = (012)\begin{pmatrix} 120100 \\ 221000 \\ 200021 \end{pmatrix} = 021012$$

ASIDE: Note that we do not in general get the messageword from the first digits of the encoded form — this only happens if G is in standard form. Indeed the digits of the messageword might not appear anywhere in the encoded version! This emphasises that the practical encoding of a message depends very much on G, rather than on C.

1.3 What is $d(C)$?

ANSWER: Clearly $d(C) \leq 3$, but no column of H is "parallel" to another, so $d(C) = 3$.

1.4 How many coset leaders are there? How many coset leaders of weight 1 are there? What are the syndromes of coset leaders?
ANSWER: $|C| = 3^3 = 27$ and $|Z_3^6| = 3^6$ so there are 27 coset leaders. Since $d(C) = 3$ all the weight 1 vectors are coset leaders. There are 12 of these. Their syndromes, and the syndrome $S(000000)$, are easy to compute:
$000000 \mapsto 000$, $x00000 \mapsto x00$ ($x \in \{1, 2\}$), $0x0000 \mapsto 0x0$,
$00x000 \mapsto xx0$, $000100 \mapsto 210$, $000200 \mapsto 120$,
$0000x0 \mapsto 00x$, $00000x \mapsto x0x$.
The remaining $27 - (12 + 1) = 14$ coset leaders are much harder to find. It is not impossible, since the standard array is not impossibly large in this case, but it is uncomfortable. In practice, a good strategy might be to wait and see what message is *received*, and hence what syndromes we need coset leaders *for* (in order to try to do error correction), rather than just computing them all up front.

Of course there are $(6.5/2)2^2 = 60$ weight 2 vectors in the space. Several, but not all, of these are in cosets led by weight 1 vectors. The syndromes of weight 2 vectors are each easy to compute by linearity, given the weight 1 syndromes above. For example:

$$S(120000) = S(100000) + S(020000) = 100 + 020 = 120 = S(000200)$$

$$S(010001) = S(010000) + S(000001) = 010 + 101 = 111 \qquad (new!)$$

$$S(001010) = S(001000) + S(000010) = 110 + 001 = 111 = S(010001)$$

But these cases illustrate the problem. The first is not new; the second is new, and can be taken as a coset leader; but the third is an equally good choice as leader of the same coset (which thus confirms that the code is not reliably 2 error correcting, as we already knew!).

To this point we do not even know if all the remaining coset leaders can be found from among the weight 2 vectors, or whether higher weights are needed. A couple more new ones at weight 2 are: $S(010010) = 011$ and $S(100001) = 201$ (and we can multiply through by 2 to get some more from these), but we would have to keep working through to find the rest. (Exercise!)

This nicely illustrates one of the problems thrown up by coding theory. The syndrome map $S : \mathbb{Z}_3^6 \to \mathbb{Z}_3^3$ is a surjective linear map. The set $\{S(e_1), S(e_2), S(e_5)\}$ is a basis of the image, so we could choose 'coset leaders' of the form $x = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_5$ with $(\alpha_1, \alpha_2, \alpha_3) \in \mathbb{Z}_3^3$, but this does *not* give the lowest possible weights, so for channels with low single digit error probability this would give highly statistically non-optimal error correction behaviour.

1.5 Given that G above is used for encoding, what messageword encodes to 212012, if any? What messageword encodes to 012212, if any?

ANSWER: encoding is

$$(x, y, z) \mapsto (x, y, z)G = (2x + y + 2z, 2x + 2y, x, y, 2z, z)$$

so for 212012 we could try to solve $2x + y + 2z = 2$, $2x + 2y = 1$, $x = 2$, $y = 0$, $2z = 1$, $z = 2$. The 3-rd, 4-th and 6-th of these give $(x, y, z) = (2, 0, 2)$ (the codeword for the letter T). The others are checks, all of which are satisfied.
For 012212 the 3-rd, 4-th and 6-th of these give $(x, y, z) = (2, 2, 2)$, but two of the checks fail, so 222 is unlikely to be what was intended!
To make a guess for the intended messageword we could compute the syndrome:

$$H(012212)^t = (2, 2, 0)^t$$

The coset leader with this syndrome is 002000. Thus the intended encoding was probably 012212-002000=010212. This decodes as 022=H.

1.6 Decode as much as possible of the following received message, given that the transmitted message was encoded using C with generator matrix G, assuming nearest neighbour decoding.

Message:

002112 012212 220112 112100 220112 000000

200021 112000 220112 000000 022022 221000

022200 000000 220112 112000 112000 101200

112000 012020 000000 221000 111112 000000

212012 010212 221000 212021 002000 211121

220112 012021 012021 200021 110221 220112

Hints:

1.6.1 The message digits in 212012 are 202 (why?)

1.6.2 202 is the representation of the 20-th letter: T.

1.6.3 The message digits in 012212 are 222. What is going on here?

Answer

ANSWER:

$002112H^t = 000$ so decode as $212 -> W$

$012212H^t = 220$ so must correct by

$012212 \to 012212 - 002000 = 010212$ so decode as $022 \to H$

$-> E \quad -> R \quad -> E \quad$ space $\quad A \quad R \quad E$

$000000 \to 000 \to$ space

$022022H^t = 111$ so must correct by some choice of weight 2 coset leader (which is at least as likely to be wrong as right, but it no worse than any other choice): choosing 010001 we get 022022 - 010001 $= 012021 ->$ $201 -> S$ (choosing 001010 we get 022022 - 001010=021012 $\to$ K here instead!)

$221000H^t = 000$ so decode as $100 \to I$

...and so on.

Hamming codes

Hamming codes

Recall: A linear [n,k]-code over F_q with PCM H has $d(C) = d$ iff every set of $d-1$ columns of H is LI, but there is a set of d columns of H that is LD.

For binary codes, no repeated column in H implies $d(C) \geq 3$. Hamming's idea was to construct the biggest possible binary H with no zero columns and no repeated columns. Fixing a positive integer r, then $\mathbb{Z}_2^r$ contains $2^r - 1$ non-zero vectors. We could simply use them all!:

0.94. Definition. Let H be a $r \times (2^r - 1)$ matrix whose columns are the distinct non-zero vectors in $\mathbb{Z}_2^r$. Then $Ham(\mathbb{Z}_2^r)$ is the binary linear code whose PCM is H.

Example

0.95. Example. For $r = 3$:

$$H = \begin{pmatrix} 0001111 \\ 0110011 \\ 1010101 \end{pmatrix}$$

Note columns ordered lexicographically. Really we think of $Ham(\mathbb{Z}_2^r)$ as a collection of several different equivalent codes, since we can order the columns as we like.

.

For $Ham(\mathbb{Z}_2^3)$ we could write

$$\tilde{H} = \left(\begin{array}{c|c} 1110 & 100 \\ 1101 & 010 \\ 1011 & 001 \end{array}\right)$$

which is in the standard form. Then the generator matrix is

$$\tilde{G} = \left(\begin{array}{c|c} 1000 & 111 \\ 0100 & 110 \\ 0010 & 101 \\ 0001 & 011 \end{array}\right)$$

.

0.96. Exercise. Connect this formulation to the example introduced earlier.

Theorem

Theorem 0.97. *$Ham(\mathbb{Z}_2^r)$ has minimum distance 3 and is perfect.*

Proof: H has no zero or parallel columns by construction, so $d \geq 3$. But it contains columns c_1, c_2, c_3 in lex order obeying $c_1 + c_2 + c_3 = 0$, so $d = 3$. Hence $Ham(\mathbb{Z}_2^r)$ is perfect iff the collection of 1-balls centred on codewords exhausts $\mathbb{Z}_2^n$, where $n = 2^r - 1$. But

$$|B_1(x)| = 1 + \binom{n}{1} = 1 + n = 2^r$$

and $M = |Ham(\mathbb{Z}_2^r)| = 2^k$ where $k = 2^r - 1 - r$. So

$$|\sqcup_{x \in Ham(\mathbb{Z}_2^r)} B_1(x)| = 2^k \times 2^r = 2^n$$

□

Hence the coset leaders of $Ham(\mathbb{Z}_2^r)$ are all vectors of weigth ≤ 1. Note that weight 1 binary vectors are just the e_is. Syndrome:

$$S(e_i) = e_i H^t = (0, 0, .., 0, 1, 0, .., 0) \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} = c_i$$

(here we write c_i for the columns written out as rows, for brevity). If the columns are ordered lexicographically then the i-th column is just the binary representation of i. So if we receive $y \in \mathbb{Z}_2^n$ with one error, its syndrome $S(y)$ is the digit position of the error (in binary).

.

0.98. Example. receive $y = 1101101$. Then

$$S(y) = (1,1,0,1,1,0,1)H^t = 101 = S(e_5)$$

Syndrome decoding: $x = y - e_5 = 1101001$.

.

Have a look for exampe here:

`https://arxiv.org/pdf/0806.2513.pdf`

for some nice results about classifying perfect binary codes of block length $2^r - 1$.

Hamming codes over non-binary fields

.

0.99. Definition. Let $u, v \in F_q^r \setminus \{0\}$. u is projectively equivalent to v, written $u \sim v$, if there exists $\lambda \in F_q \setminus \{0\}$ such that $u = \lambda v$. This says that u, v are parallel. (NB, being parallel is an equivalence relation.)
We call the set of projective equivalence classes the projective space of F_q^n, denoted $P(F_q^n)$.

0.100. Example. $\mathbb{Z}_5^2$ has the following projective equivalence classes:

$$[01] = \{01, 02, 03, 04\}$$

$$[10] = \{10, 20, 30, 40\}$$

$$[11] = \{11, 22, 33, 44\}$$

$$[12] = \{12, 24, 31, 43\}$$

...

$$[14] = \{14, 23, 32, 41\}$$

In general there are $q - 1$ elements in each class, so there are $\frac{q^r-1}{q-1}$ projective equivalence classes.

.

0.101. Definition. Let H be a $r \times \frac{q^r-1}{q-1}$ matrix (over F_q) each of whose columns belongs to a different class in $P(F_q^r)$. Then the q-ary linear code whose PCM is H is a q-ary Hamming code, denoted $Ham(F_q^r)$.

0.102. Example. For $Ham(F_5^2)$ we could choose PCM

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 \end{pmatrix} \qquad or \qquad H' = \begin{pmatrix} 0 & 3 & 4 & 1 & 2 & 3 \\ 2 & 0 & 4 & 2 & 1 & 2 \end{pmatrix}$$

Of these, H is best for easy decoding in practice. In H we chose from each class the unique vector whose first non-zero digit is 1; and then ordered the vectors lexicographically. (If we refer to *the* code $Ham(F_5^2)$, this is the PCM we mean.)

Theorem 0.103. *$Ham(F_q^r)$ has minimum distance 3 and is perfect.*

Proof: Exercise (optional).

SYNDROME DECODING:

Again we know that coset leaders are vectors of weight ≤ 1, that is, the zero vector (let's call it $\underline{0}$); and the vectors of form Ae_i, where $A \in F_q \setminus \{0\}$ and $1 \leq i \leq n$.

Syndromes: $S(\underline{0}) = \underline{0}$

$$S(Ae_i) = Ae_iH^T = A[0, 0, .., 0, 1, 0, .., 0] \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_i \\ \vdots \\ c_n \end{pmatrix} = Ac_i$$

NB, if H is our 'standard' choice , then
first non-zero digit of c_i is 1 implies first non-zero digit of $S(Ae_i)$ is A, which implies that we can read off A immediately.

SCHEME: (i) receive y; compute $S(y)$;
(ii) If $S(y) = 0$ then $x = y$;
(iii) any other $S(y) \in F_q^r$ must lie in one of the classes of $P(F_q^r)$, so $S(y) = Ac_i = S(Ae_i)$ for some $A \in F_q \setminus \{0\}$, $1 \leq i \leq n$.
Decode by subtracting A from digit i:

$$y \mapsto x = y - Ae_i$$

0.104. Example. $Ham(F_4^2)$

$n = |P(F_4^2)| = \frac{4^2-1}{4-1} = 5$, $r = 2$ implies $k = 3$, so we have a $[5, 3, 3]$-code over F_4.

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

Suppose we receive $y = bab10$. We have

$$S(y) = [b, a, b, 1, 0]H^T = [a + b + 1 + 0, b + b + a]$$

$$= [1 + 1, a] = [0, a] = a[0, 1] = ac_1 = S(ae_1)$$

so

$$y \mapsto x = y - ae_i = [b - a, a, b, 1, 0] = 1ab10.$$

In summary, this is very similar to previous examples. The main change is in the type of arithmetic done.

Cyclic codes

Definition

0.105. Definition. A code C is cyclic if it is linear and any cyclic shift of a codeword is also a codeword.

0.106. Example. 2-ary code $C = \{000, 101, 011, 110\}$ is cyclic.

We continue this section by introducing the technology we shall need. The use we shall make of it comes later.

0.107. Definition. Let F be a field. Then $F[x]$ is the set of all polynomials in x:

$$a(x) = \sum_i a_i x^i$$

where $a_i \in F$. If $a(x)$ has degree m and $a_m = 1$ then $a(x)$ is said to be monic.

$F[x]$ is a ring, but not a field.

Associated to any polynomial $a(x) \in F[x]$ there is a function: $x \mapsto a(x)$ (the evaluation function). In general a polynomial is more than a function, however, in the following sense.

Example

0.108. Example. There are 4 distinct functions from $\mathbb{Z}_2 \to \mathbb{Z}_2$. But there are infinitely many different polynomials in $\mathbb{Z}_2[x]$. E.g. $a(x) = x^5 + x^2 + x + 1$, $b(x) = x^{17} + 1$, both have the same function associated to them (exercise!).

Remainder Theorem

Theorem 0.109. *[The remainder theorem]*
For every pair $a(x), b(x) \in F[x]$ *with* $b(x) \neq 0$, *there exists a unique pair* $q(x)$ *(the quotient) and* $r(x)$ *(the remainder) in* $F[x]$ *such that* $deg(r(x)) < deg(b(x))$ *and* $a(x) = q(x)b(x) + r(x)$.
Proof: Can construct $q(x)$, $r(x)$ by usual long-division algorithm, using appropriate arithmetic. □

0.110. Exercise. Divide $a(x) = x^3 + 3x^2 + 4$ by $b(x) = 2x^2 + 3$ in $\mathbb{Z}_5[x]$. [10]

[10] Answer:

$$x^3 + 3x^2 + 4 = (3x + 4)(2x^2 + 3) + (x + 2)$$

0.111. Definition. Choose a *fixed* polynomial $f(x) \in F[x]$. Then polynomials $a(x), b(x) \in F[x]$ are *congruent modulo* $f(x)$ (written $a(x) \equiv b(x)$ mod. $f(x)$), if $a(x) - b(x)$ is divisible by $f(x)$ (meaning $a(x) - b(x) = q(x)f(x)$ for some $q(x) \in F[x]$, with no remainder).

This is an *equivalence relation* on $F[x]$ (check it! This is just like our modular arithmetic).

As usual, denote equivalence (congruence) class of $a(x)$ by

$$[a(x)] = \{b(x) \in F[x] \mid b(x) \equiv a(x) \ \ mod.f(x)\}$$

Let $F[x]/f(x)$ denote the set of such classes. We can define addition and multiplication on $F[x]/f(x)$:

$$[a(x)] + [b(x)] = [a(x) + b(x)]$$

$$[a(x)][b(x)] = [a(x)b(x)]$$

(These are well defined by a lemma that you should state and check, analogous to one we had earlier.)

By these operations $F[x]/f(x)$ is a ring.

Any polynomial $a(x) \in F[x]$ has a unique remainder $r(x)$ 'modulo' $f(x)$, with $deg(r(x)) < deg(f(x))$ by Theorem 0.109.

0.112. Lemma. *$a(x) \equiv a'(x)$ mod. $f(x)$ iff their remainders $r(x), r'(x)$ are equal.*

The upshot of this is that we can identify $[a(x)]$ with $r(x)$, the remainder of any of its elements. In this way we may identify

$$F[x]/f(x) \leftrightarrow \{\sum_{i=0}^{n-1} a_i x^i \mid a_0, a_1, .., a_{n-1} \in F\}$$

the set of polynomials of degree $< \ deg(f(x)) = n$. Of course, this set may then be identified with F^n — the list of coefficients.

Altogether this gives us a way to regard the vector space F^n as a ring. That is, we equip it with the extra operation of multiplication of vectors!

.

0.113. Example. $R = \mathbb{Z}_2[x]/(x^2 + x + 1)$ (NB $f(x)$ here has degree 2), gives $R \equiv \{0, 1, x, 1 + x\}$ = polynomials of degree < 2.
Compute the addition and multiplication tables.
Can you compute inverses too?

In fact every non-zero element does have an inverse, so R is even a field in this case!

0.114. Definition. $f(x) \in F[x]$ is reducible if there exist $a(x), b(x) \in F[x]$ with degrees less than that of $f(x)$, such that $f(x) = a(x)b(x)$.

FACT: $F[x]/f(x)$ is a field iff $f(x)$ is not reducible (irreducible).

.

0.115. Lemma. *(i) $f(x) \in F[x]$ has a degree 1 factor $(x-a)$ iff $f(a) = 0$.*
(ii) If degree $f(x)$ =2 or 3 then $f(x)$ is irreducible iff for all $a \in F$, $f(a) \neq 0$.
(iii) Over any field F, $x^n - 1 = (x-1)(x^{n-1} + x^{n-2} + ... + x + 1)$.
Proof: (i) Use Theorem 0.109. (iii) by induction on n. $\square$

0.116. Example. Completely factorise $x^4 - 1 \in \mathbb{Z}_5[x]$. [11]

[11] Answer: over $\mathbb{Z}_5$

$$x^4 - 1 = (x-1)(x-2)(x-3)(x+1) = (x+4)(x+3)(x+2)(x+1)$$

.

For cyclic codes the ring of interest is as follows.

0.117. Definition. For given field F, define

$$R_n = F[x]/(x^n - 1)$$

NOTES

NOTES:

(a) $(x^n - 1)$ is always reducible, so R_n is never a field.

(b) $x^n \equiv 1$ mod. $x^n - 1$, so $x^{n+m} = x^m$ for any m. No need to use remainder theorem to compute products.

E.g. in $R_5 = \mathbb{Z}_3[x]/(x^5 - 1)$

$$(x^2+x)(x^4+2) = x^6+2x^2+x^5+2x \equiv x+2x^2+1+2x = 2x^2+1$$

(c) Since $deg(x^n - 1) = n$ we can identify R_n with polynomials of degree less than n, and hence with F^n:

$a_0 + a_1x + ... + a_{n-1}x^{n-1} \leftrightarrow (a_0, a_1, .., a_{n-1})$

addition of polys $\leftrightarrow$ vector addition

multiplication by constant $\leftrightarrow$ scalar multiplication

multiplication by $x \leftrightarrow$ cyclic shift.

Back to the codes

We can think of a q-ary code of block-length n as a subset of R_n (with $F = F_q$). Then:

Theorem 0.118. *A code $C \subset R_n$ is a cyclic code iff*
(i) $a(x), b(x) \in C$ implies $a(x) + b(x) \in C$;
(ii) $a(x) \in C$, $r(x) \in R_n$ implies $r(x)a(x) \in C$.
(NB (ii) is more than closure of C under multiplication!)

0.119. Definition. Let $f(x) \in R_n$. Then

$$\langle f(x) \rangle \;=\; \{r(x)f(x) \mid r(x) \in R_n\}$$

called "the ring span of $f(x)$".
Clearly this satisfies properties (i), (ii) of Theorem 0.118. Hence it is a cyclic code over F_q — the cyclic code generated by $f(x)$.

0.120. Example. $F_q = \mathbb{Z}_2$, $C = \langle 1 + x^2 \rangle \subset R_3 = \mathbb{Z}_2[x]/(x^3 - 1)$

$$R_3 = \{0, 1, x, 1 + x, x^2, 1 + x^2, x + x^2, 1 + x + x^2\}$$

$$C = \{0, 1+x^2, x+1, x^2+x, 1+x, x+x^2+x^3+x^4, 1+x+x^2+x^2+x^3+x^4\}$$

$$= \{0, 1 + x^2, 1 + x, x + x^2\} \leftrightarrow \{000, 101, 110, 011\} \subset \mathbb{Z}_2^3$$

0.121. Exercise. Show that $\langle 1 + x^2 \rangle = \langle 1 + x \rangle = \langle x + x^2 \rangle$ in this case.

That is, more than one polynomial can generate a given cyclic code. However, there is a canonical choice of generating polynomial:

Theorem 0.122. *Let C be a non-zero cyclic code in R_n. Then*
(i) there exists a unique monic polynomial $g(x)$ of least degree in C;
(ii) $C = \langle g(x) \rangle$. In fact every codeword $a(x) \in C$ is a strict *multiple of $g(x)$: $a(x) = r(x)g(x)$ (not just congruent mod. $x^n - 1$).*
(iii) $g(x)$ is a factor of $x^n - 1$.

0.123. Definition. The unique minimal degree monic polynomial $g(x)$ in a cyclic code C is called the generator polynomial of C.

For example, $g(x) = 1 + x$ is the gen. poly. for our last example.

CRUCIAL FACT:

Since the gen. poly. is unique, cyclic codes of length n are in 1-to-1 correspondence with monic factors of $x^n - 1$. This completely characterises all cyclic codes!

0.124. Example. Find all cyclic codes over $\mathbb{Z}_2$ of length 3.
$C \subset R_3 = \mathbb{Z}_2[x]/(x^3 - 1)$. But

$$x^3 - 1 = (x - 1)(x^2 + x + 1) = (x + 1)(x^2 + x + 1)$$

so there are four such codes: $\langle 1 \rangle = R_3 = \mathbb{Z}_2^3$
$\langle 1 + x \rangle = \{0, 1 + x, 1 + x^2, x + x^2\} = \{000, 110, 101, 011\}$
$\langle 1 + x + x^2 \rangle = \{0, 1 + x + x^2\} = \{000, 111\}$
$\langle (1 + x)(1 + x + x^2) \rangle = \{0\} = \{000\}$

0.125. Example. How many cyclic codes of length 4 over $\mathbb{Z}_5$ are there?

Answer: same as number of monic factors of $x^4 - 1 \in \mathbb{Z}_5[x]$. But we already saw that $x^4 - 1 = (x+4)(x+3)(x+2)(x+1)$ over $\mathbb{Z}_5$, so the general monic factor is

$$g(x) = (x+4)^{p_4}(x+3)^{p_3}(x+2)^{p_2}(x+1)^{p_1}$$

where each p_i can be either 0 or 1. Since there are 2^4 choices here, we have 16 cyclic codes.

For example $p_1 = p_4 = 1$, $p_2 = p_3 = 0$ gives code $\langle (x+1)(x+4) \rangle = \langle x^2 - 1 \rangle$

What can we say about this code? What is its dimension?

Theorem 0.126. *Let $g(x) = \sum_{i=0}^{r} g_i x^i$ be the gen. poly. for cyclic code C (note $g_r = 1$). Then*

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & .. & g_r & 0 & .. & 0 \\ 0 & g_0 & g_1 & g_2 & .. & g_r & .. & 0 \\ & & \ddots & & & & & \\ 0 & ... & 0 & g_0 & g_1 & g_2 & .. & g_r \end{pmatrix}$$

is a generator matrix for C.

Note that this G is a $(n-r) \times n$ matrix.

0.127. Corollary. *A cyclic code $C \subset R_n$ whose gen. poly. has order r has dim. $k = n - r$ and has redundancy r. (Cf. Def.0.82.)*

0.128. Example. Construct a generator matrix for each 3-ary cyclic code of length 4.

$R_4 = \mathbb{Z}_3[x]/(x^4 - 1)$

$$(x^4-1) = (x^2-1)(x^2+1) = (x-1)(x+1)(x^2+1) = (x+1)(x+2)(x^2+1)$$

(NB $(x^2 + 1)$ is irreducible here) so there are 2^3 monic factors, hence 8 cyclic codes generated by

$$g(x) = (x + 1)^{p_1}(x + 2)^{p_2}(x^2 + 1)^{p_3}$$

with $p_i \in \{0, 1\}$. We have the list given in Table 2.

$g(x)$	*redundancy*	*dimension*	G
1	0	4	1_4
$1+x$	1	3	$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$
$2+x$	1	3	$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{pmatrix}$
$1+x^2$	2	2	$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$
$(1+x)(2+x)$	2	2	$\begin{pmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}$
$(1+x)(1+x^2)$	3	1	$\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$
$(2+x)(1+x^2)$	3	1	$\begin{pmatrix} 2 & 1 & 2 & 1 \end{pmatrix}$
$(1+x)(2+x)(1+x^2)$	4	0	$-$

Table 2:

To find $d(C)$, do syndrome decoding, etc, it is better to have a PCM than a generator matrix. So how can we construct H here?

Recall H is generator matrix for $C^{\perp}$.

Theorem 0.129. *If C is cyclic so is $C^{\perp}$.*

So $C^{\perp}$ has a unique generator polyomial, that is also a factor of $x^n - 1$. If we find it we can use Theorem 0.126 to find a generator matrix for $C^{\perp}$ and hence a PCM for C.

0.130. Definition. Let $C \subset R_n$ be cyclic with generator polynomial $g(x)$. Then Theorem 0.122 implies that there exists another polynomial $h(x)$ such that $x^n - 1 = g(x)h(x)$, and $h(x)$ is unique by Theorem 0.109. We call $h(x)$ the *check polynomial* of the code C.

0.131. Example. Given that $g(x) = x^2 + x + 3$ is the gen. poly. of a cyclic 5-ary [4,2]-code C, we have

$$(x^2 + x + 3)(x^2 + 4x + 3) \equiv x^4 - 1$$

so $h(x) = (x^2 + 4x + 3)$.
Note incidentally that $C^\perp \neq \langle h(x) \rangle$.

Theorem 0.132. *Let $h(x)$ be the check poly. for code C. Then $a(x) \in C$ iff $a(x)h(x) \equiv 0$.*

It is not true in general that $C^\perp = \langle h(x) \rangle$, but we can construct the gen. poly. for $C^\perp$ from $h(x)$.

Define

$$H = \begin{pmatrix} h_k & h_{k-1} & h_{k-2} & .. & h_0 & 0 & .. & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & .. & h_0 & .. & 0 \\ & & \ddots & & & & & \\ 0 & ... & 0 & h_k & h_{k-1} & h_{k-2} & .. & h_0 \end{pmatrix}$$

We have

$$\begin{pmatrix} a_0 & a_1 & ... & a_{n-1} \end{pmatrix} H^T = 0 \qquad (3)$$

for $a(x) \in C$.

Consider C', code generated by H. Since $h(x)$ is monic we have $h_k = 1$, so leading diagonal is all 1s. Thus H has maximal rank $((n-k))$, so $dim\ C' = n-k$. Also, any $\underline{w} \in C'$ is perpendicular to all $a(x) \in C$ by (3). Thus $C' \subset C^{\perp}$. But $dim\ C' = dim\ C^{\perp}$, so $C' = C^{\perp}$.

Theorem 0.133. *Let $C \subset R_n$ be a cyclic code with check poly. $h(x)$. Then H is a PCM for C.*

0.134. Example. Recall $h(x) = 3 + 4x + x^2$ is check poly. for $C = \langle x^2 + x + 3 \rangle \subset R_4 = \mathbb{Z}_5[x]/(x^4 - 1)$. Hence

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

is a PCM for C.
Exercise: check $aH^T = 0$ for all $a \in C$.

THIS is what we want! A construction for the PCM for C. Armed with this, we can do our usual routines for coding with C.

It remains to compute a gen poly. for $C^\perp$:

Comparing G and H we see that they are of similar form. However the reversing of the indices means that it is an open question whether

$$g^{\perp}(x) = h_k + h_{k-1}x + ... + h_0x^k$$

— the candidate for the gen. poly. for $C^{\perp}$ on this basis — is monic.

We can obtain a monic version by dividing by h_0... ...unless $h_0 = 0$. But we need not worry about this: If $h_0 = 0$ then $h(0) = 0$ and

$$x^n - 1 = g(x)h(x) \quad \Rightarrow \qquad -1 = g(0)h(0) = 0$$

which cannot happen.

0.135. Definition. Given $p(x) = p_0 + p_1x + ... + p_kx^k \in F_q[x]$ ($p_k \neq 0$) the *reciprocal* of $p(x)$ is

$$\overline{p}(x) = p_k + p_{k-1}x + ... + p_0x^k \in F_q[x]$$

So we have

0.136. Corollary. *Let $C \subset R_n$ be a cyclic code with check poly. $h(x)$. Then $C^\perp$ is the cyclic code with gen poly.*

$$g^\perp(x) = h(0)^{-1}\overline{h}(x)$$

0.137. Example. $C = \langle x^2 + x + 3 \rangle \subset R_4 = \mathbb{Z}_5[x]/(x^4 - 1)$ has check poly $h(x) = 3 + 4x + x^2$. Hence gen poly. for $C^\perp$ is

$$g^\perp = h(0)^{-1}\overline{h}(x) = 3^{-1}(1+4x+3x^2) = 2(1+4x+3x^2) = 2+3x+x^2$$

0.138. Exercise. Construct a generator matrix for the binary $n = 7$ cyclic code with generator polynomial $1 + x + x^3$. What can you say about it?

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

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

0.139. Exercise. Let C, C' be cyclic codes. What can you say about their intersection?

0.140. Exercise. If C is cyclic and C' is an equivalent code, is C' necessarily cyclic?

Golay codes

Golay

0.141. Exercise. Have a go at verifying that

$$x^{23}-1 = (x-1)(x^{11}+x^{10}+x^6+x^5+x^4+x^2+1)(x^{11}+x^9+x^7+x^6+x^5+x+1)$$

is the irreducible factorisation over F_2. (This is not easy. But it is interesting to have a go. See later for some hints.)

The cyclic code generated by either of the big factors is the 'Golay code' G_{23}. (Exercise: Strictly speaking, the 'Golay code' refers to the equivalence class of codes. Show that the two factors give equivalent but not identical codes. See later for hints.)

The extension of this code by a parity check bit is the 'extended Golay code' G_{24}.

Theorem: a) G_{24} is self-dual. b) The weight of every codeword in G_{24} is a multiple of 4. c) But it has no codeword of weight 4 so min distance is 8.

Proof ideas: a) The easiest thing is to write out a generator matrix and do some checking. Here we go!... First a generator matrix for G_{23}:

$$M_1 = \begin{pmatrix} 10101110001100000000000 \\ 01010111000110000000000 \\ 00101011100011000000000 \\ 00010101110001100000000 \\ 00001010111000110000000 \\ 00000101011100011000000 \\ 00000010101110001100000 \\ 00000001010111000110000 \\ 00000000101011100011000 \\ 00000000010101110001100 \\ 00000000001010111000110 \\ 00000000000101011100011 \end{pmatrix}$$

Since the row vectors in M_1 are all weight 7, those in the extension to G_{24} will have weight 8.

Let's compute $r_1.r_2 = 3$ here, so $= 4 = 0$ for G_{24}. Obviously then $r_i.r_{i+1} = 3$ for any i.

What about $r_i.r_{i+2}$?

...And so on. :-)

Having shown that the vectors are pairwise orthogonal it remains only to show that there are enough of them... Hint: $24 - 12 = 12$.

b) Let's stick with G_{24}. All the rows have weight 8. What about $r_1 + r_2$ etc? What is a formula for weight here? Hint: use duality.

c) How could a codeword of weight 4 arise? Something to think about! :-)

Remark: The hope here is to finish the module with some activities of a more research-like nature. The codes we have studied recently lend themselves well to this. But the acitivites themselves are probably best conducted by discussion rather than by formal lecturing to lecture notes. So,...

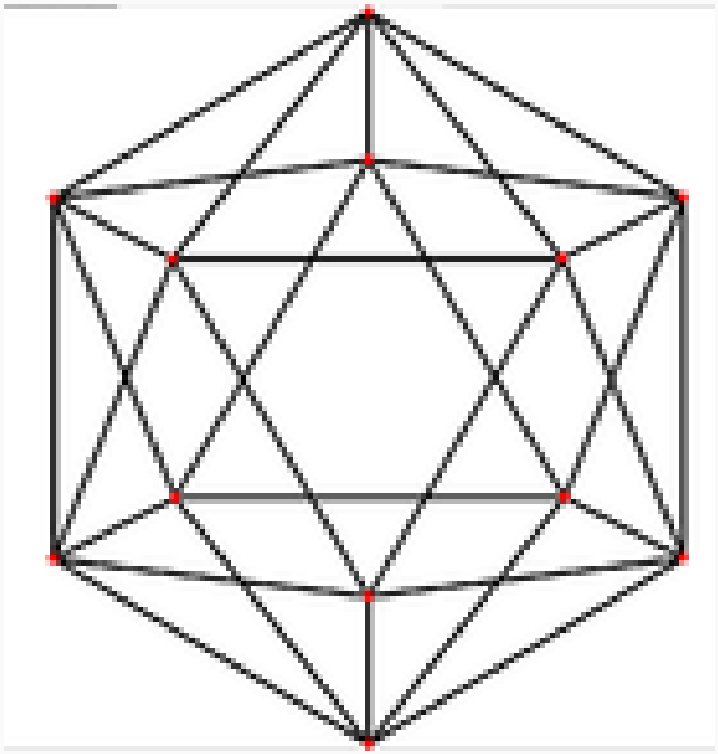
Golay code experiments

Investigate whether the Golay code G_{24} has standard form $G_A = [1_{12}|A]$ where

$$A = \begin{pmatrix} 011111111111 \\ 111011100010 \\ 110111000101 \\ 101110001011 \\ 111100010110 \\ 111000101101 \\ 110001011011 \\ 100010110111 \\ 100101101110 \\ 101011011100 \\ 110110111000 \\ 101101110001 \end{pmatrix}$$

Note that $H = [A|1_{12}]$. (Think about it!)

Aside on the Icosahedron



— compute the complement of the adjacency matrix.

— this figure has a lot of symmetry. (E.g. rotations order 2,3,5.)

— So what, cf. generator matrices!? LOL. (No order 11.)

Back to Theorem

(b): need more clues? Think about wt(r1+r2)=wt(r1)+wt(r2)-something!

(c): Suppose for a contradiction that G_{24} contains a codeword with $wt(v) = 4$.

Write v as $(v1|v2)$, where $v1$ is the first 12 bits of v, and $v2$ is the last 12 bits of v.

Every codeword is a 2-ary linear combination of rows from a generator matrix, and hence is a sum of a subset of rows. If we fix a standard generator matrix (which we know exists, even if it is not G_A above) then wt(v1) gives the number of rows involved in v.

Since wt(v) = wt(v1) + wt(v2), one of the following must hold: wt(v1) = 0. This cannot happen since the only such word is 0, which is weight 0.

...

...

wt(v1) = 1. Again looking at standard G, v must be one of the rows of G. Now suppose G_A above is indeed a generator matrix. — no wt(v2)=3 occurs there, so we get a contradiction.

wt(v2) = 2. Then v is the sum of two rows of G. We should CHECK (see below)

that none of such give wt(v2) = 2.

wt(v1) = 3 and wt(v2) = 1. Since H is a generator matrix, wt(v2)=1 says that v must be one of the rows of H. This gives a contradiction.

wt(v1) = 4 and wt(v2) = 0. This is similar to case 1, using H instead of standard G.

DONE.

Completing the CHECK we skipped

Back with establishing a contradiction for wt(v2)=2, we are asking to check that this can never happen for v a sum of two rows.

Thus we need to inspect $r_i + r_j$ for all i, j. This is not impossible to do by brute force, but we can be cleverer. Note that the 'v2 part' of the generator matrix is A itself. Note that

$$A = \begin{pmatrix} 0 & 11...1 \\ (11...1)^t & B \end{pmatrix}$$

where B has the cyclic property.

By this observation it is enough to check the cases $r_1 + r_j$ and $r_2 + r_j$ only.

Since $1 + 1 = 0$ then $wt(r_i + r_j)$ just counts the number of places where the two rows differ. So we get $wt(r_1 + r_2) = 2 + 6$ (splitting into the v1 and v2 parts).

Since all the other r_js are obtained by cyclic shifting in the last 11 positions, but this does not affect r_1, we get $wt(r_1 + r_j) = 2 + 6$.

It only remains to check $r_2 + r_j$ for the other js — 10 easy checks. For example $wt(r_2 + r_3) = 2 + 6$. (DONE.)

And finally, what about the generator matrix supposition?

There is a lot of mathematical fun to be had with this. Here we just make some remarks.

1. Since the Golay code is defined up to equivalence, the supposition is not necessarily that we can get from M_1 to $[1_{12}|A]$ by elementary column operations. We may use elementary row operations as well.

2. To get warmed up, let's show that M_1 and M_2 (the version obtained from the other big factor in $x^{23} - 1$) give equivalent but not identical codes.

Warm up

First the codes are not identical: let's write EM_1 for M_1 with the parity check column; and EM_2 similarly. If M_1 and M_2 give identical codes, then they have the same parity check extension code, so it is enough to compare the extended versions.

Now consider $r_1.\rho_1$, where r_i is the ith row of EM_1 and ρ_i is the ith row of EM_2. The matrices EM_1, EM_2 start:

$101011100011|0000000000001$

$110001110101|0000000000001$

We see that these rows are not orthogonal. But G_{24} is self-dual. So ρ_1 cannot lie in the code generated by EM_1.

But the codes are equivalent: consider reversing all rows and columns of M_2 (noting that the polynomials are 'reverses' of each other) ...

For completeness here is

$$EM_1 = \begin{pmatrix} 101011100011000000000001 \\ 010101110001100000000001 \\ 001010111000110000000001 \\ 000101011100011000000001 \\ 000010101110001100000001 \\ 000001010111000110000001 \\ 000000101011100011000001 \\ 000000010101110001100001 \\ 000000001010111000110001 \\ 000000000101011100011001 \\ 000000000010101110001101 \\ 000000000001010111000111 \end{pmatrix}$$

Remarks continued

3. Try the same games comparing EM_1 (and then EM_2) with $[1_{12}|A]$.

Cases: (here r_i is row i of EM_1 and α_i is row i of $[1_{12}|A]$)

$r_1.\alpha_1 = 0$

$r_1.\alpha_2 = 0$

$r_1.\alpha_3 = 0$

$r_1.\alpha_4 =?$

...

4. Still intrigued? Have a look in Conway and Sloane's book on sphere packings.

Perfection

Is G_{23} perfect?

We are asking if

$$\sum_{i=0}^{3} \binom{23}{i} = \frac{2^n}{2^k} = 2^{11} = 2048$$

We get $1 + 23 + \frac{23.22}{2} + \frac{23.22.21}{2.3} = 1 + 23 + 253 + 1771 = ...$

...Now think about this question more generally! (As Golay did.)

A related question: for what values of $[n, k, 3]$ is it 'numerically possible' to have a perfect (binary) code? What do you notice about the answer?

Gauss and irreducible monic polynomials

Gauss has a useful result to help us answer the question at the start of this section about irreducible polynomials.

Gauss showed that for $n = p$ prime, the product of all irreducible monic polynomials in $F_2[x]$ of order n is given by a simply expressed (big) polynomial $\Gamma(x)$:

$$\Gamma(x) = \frac{x^{2^p} - x}{x^2 - x}$$

This means that we can check if a polynomial of order $n = p$ is irreducible by checking if it divides $\Gamma(x)$. (See the online references for more details.)

...

Ternary Golay code

The code C_{12} generated by

$$G_{12} = \begin{pmatrix} 100000011111 \\ 010000101221 \\ 001000110122 \\ 000100121012 \\ 000010122101 \\ 000001112210 \end{pmatrix}$$

is the extended ternary (q = 3) Golay [12, 6, 6]-code.

(a) what is $r_i.r_j$ for G_{12} here? find a parity-check matrix.

(b) Decode the received vector $y = 010000010101$, assuming that at most two errors have occurred.

(c) factorise $x^{11} - 1$ over F_3. (Hint: it is something to do with this code!)

(d) what else can you say about this code?

Some answers:

(a) $r_1.r_1 = 6 \equiv 0$
$r_1.r_2 = 6 \equiv 0$
$r_2.r_2 = 12 \equiv 0$
and so on. On the other hand $12 - 6 = 6$, so this is its own PCM.

(b) a syndrome is

$$Gy^t = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

We can take the columns of G_{12} as the syndromes of the coset leaders. Thus the error is composed of an error bit in position 7 and 2 error bits in position 6. That is, the error vector is $e = 000002100000$, so the transmitted vector was $x = y - e = 010001210101$.

That's enough new material for now

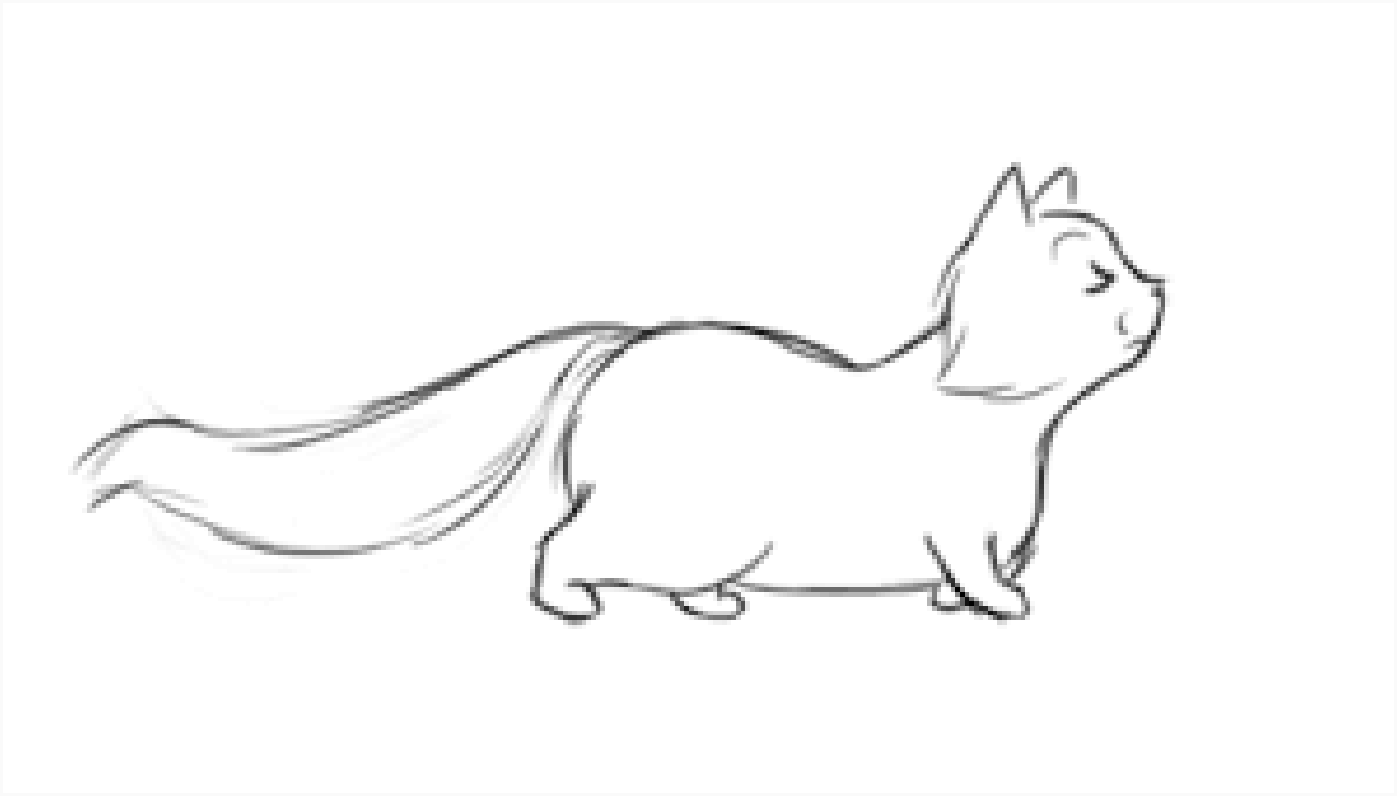


Figure 6: Cuthberta's evident self-confidence stemmed from the fact that, during her semester trapped in The Matrix, she had in fact learned a considerable amount about ...cyclic codes.

Yet More Exercises

0.142. Exercise. Can you describe an 'alphabet' Σ_q of size q, and give an n such that this entire *question* is a codeword in some $C \subset \Sigma_q^n$?
Answer:

If we have a Σ_q consisting of all upper and lower case letters, all Greek letters, all punctation, some typesetting instructions (subscript etc), and a 'space' symbol, then we can assemble the question from these. The q is roughly 52+20+20+20 (say). Let's add in some math symbols too, and say $q = 130$.
For n we just count up the number of symbols in the question, including spaces etc: roughly $n = 100$.

Another construction would be to have a Σ_q containing highly complex composite symbols, whose shapes form whole words, or perhaps even whole sentences. Of course this is much less realistic (the symbol set would be difficult or impossible to use in most circumstances), but in the extreme we could have $q = 1$ (the element is an image of the whole question) and $n = 1$!

0.143. Exercise. A code C is known to be 21 error correcting. State a lower bound on $d(C)$.
Answer:

By our Proposition: If $d(C) \geq 2t + 1$ then C can correct up to t errors by the 'pick the closest' strategy.
Thus in our case $d(C) \geq 43$.

0.144. Exercise. The set E_n of even weight binary vectors of length n is a subspace of $\mathbb{Z}_2^n$. Hence E_n is a binary linear code. What are the parameters $[n, k, d]$ of E_n? Write down a generator matrix for E_n in standard form.

Answer:

$k = dim\ E_n = n - 1$.

d is the minimum weight of non-zero vectors in E_n , which must be 2 (all vectors have even weight, so $d \geq 2$, and $110...0 \in E_n$ has weight 2). Hence E_n is a binary linear $[n, n-1, 2]$-code. Its generator matrix in standard form is

$$\begin{pmatrix} 100...01 \\ 010...01 \\ 001...01 \\ ..... \\ 000...11 \end{pmatrix}$$

0.145. Exercise. (i) Construct a binary linear [8,4,3]-code.
(ii) How many different matrices in standard form generate such codes?
Answer:

(i) We are looking for a length 8 code $C \subset \mathbb{Z}_2^8$, with dimension 4, thus we are looking for a 4×8 generator matrix. Putting this in standard form (without loss of generality) means

$$G = [1_4|A]$$

where A is a 4×4 binary matrix. There are a grand total of $2^{4\times 4}$ such matrices, including the zero matrix; the identity matrix and so on. Not all of them generate $d = 3$ codes, however. For example $A = 0$ means that each generating vector (in $G = [1_4|0]$) has total weight 1 (so $d = 1$). Similarly $A = 1_4$ means that each generating vector (in $G = [1_4|1_4]$) has total weight 2.

In other words each row of A must have at least two non-zero entries (so that each row of G has at least three), if we want $d = 3$. For example

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

Now for each candidate, such as this, we need to check that the minimum weight of all nonzero vectors is 3. There are several ways to do this. One way is to construct the PCM:

$$H = [-A^t | 1_4] = [A^t | 1_4]$$

Here we just need to check that no column is zero and no two columns are the same (by Theorem 0.89). This is clearly the same as checking that no two *rows* of A are the same (we have

stipulated that each row has at least two non-zero entries, so their transposes cannot be the same as any of the vectors in 1_4).
This requirement is satisfied in our example, so we are done. The code in full consists in all linear combinations of the row vectors in our G. Thus it starts

$$C = \{00000000, 10001100, 01001010, 00101001, 00010011, 11000110, 101$$

$$..., 11111100\}$$

(16 elements altogether).

Remark: Another example satisfying the criteria is give by

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

Remark: $A_2(8,3) = 20$, so a binary (8,20,3)-code exists. Of course no such code can be linear, since $|C| = M = q^k$ for some integer dimension k for a linear code, and no integer k obeys $20 = 2^k$! Thus $|C| = 2^4 = 16$ is the biggest *linear* code we can hope for.

(ii) As for the number of such distinct generating matrices $G = [1_4|A]$, we can choose the first row of A freely, except not choosing 0000, or any of the four weight-1 vectors, so there are $2^4 - 5$ possibilities. The second row can be chosen freely except not 0000, or weight-1, and not the same as the first row, so there are $2^4 - 6$ possibilities. Continuing similarly, altogether we have $\frac{(2^4-5)!}{(2^4-9)!}$ choices.

0.146. Exercise. (i) Construct standard arrays for the binary

linear codes C_1, C_2 generated by

$$G_1 = \begin{pmatrix} 101 \\ 011 \end{pmatrix} \qquad G_1 = \begin{pmatrix} 10110 \\ 01011 \end{pmatrix}$$

(ii) Decode the received vectors 11111 and 11011 and 01011 in C_2.
(iii) We receive 00101. What is going on?! Explain with a ball-packing analogy.

Answer:
(i) $C_1 = \{000, 101, 011, 110\}$. Evidently this has $d(C_1) = 2$.
As usual, the first row of the array is

$$000 \;\; 101 \;\; 011 \;\; 110$$

Since 100 is not in the code, and hence has not appeared in the array so far, we can use it as the next coset leader (we could have used 001 instead, say). We get the next row:

$$100 \;\; 001 \;\; 111 \;\; 010$$

(just vector shift all the code elements by 100).
At this point we see that all eight vectors in $\mathbb{Z}_2^3$ are IN, so we stop. Since the weight 1 vectors are not in distinct rows, this code is not even 1 error correcting. (Of course we knew that already, since $d(C_1) < 3$.)

For C_2 we start with the code itself in the first row:

$$00000 \;\; 10110 \;\; 01011 \;\; 11101$$

Then we construct rows with coset leaders which are (a) not in the code; (b) of lowest possible weight, i.e. of weight 1. Since 10000 has not appeared in the code we lead with that next:

$$10000 \;\; 00110 \;\; 11011 \;\; 01101$$

then

$$01000 \;\; 11110 \;\; 00011 \;\; 10101$$

In this case none of the weight 1 vectors appear in each other's cosets, so we construct a total of 5 rows this way, continuing with coset leaders 00100, 00010 and 00001. E.g.

$$00010 \quad 10100 \quad 01001 \quad 11111$$

Since $|\mathbb{Z}_2^5| = 32$ and each row has 4 vectors in it, we need 8 rows altogether. We have 6 so far. The remainder will have to be led by vectors of higher weight.
Some weight 2 vectors have already appeared, but 11000, 10001, 00101, 01100 have not. We can make:

$$11000 \quad 01110 \quad 10011 \quad 00101$$

and

$$10001 \quad 00111 \quad 11010 \quad 01100$$

and then we are done.

Of course we could have started with one of the others, which would have produced a different array! This tells us that our code is not reliably 2 error correcting.

(ii) 11111 lives in a column below codeword 11101

11011 lives in a column below codeword 01011

01011 is a codeword.

(iii) We decode 00101 as 11101, because we decide that the error in this case was 11000, based on our array (11000 is the coset leader). Clearly there is no codeword closer to 00101 than 11101, but there *are* codewords equidistant. Indeed 00000 is a codeword at distance 2 from 00101. Statistically, then, we might just as well have decoded 00101 as 00000 — and that is what we *would* have done if we had made a different arbitrary choice of a weight 2 coset leader.

This just goes to show that our error correction is not perfect. It is

just the best we can do.

In ball-packing terms, the ball around 00000 of 'radius' 2 intersects the ball around 11101 of radius 2. They intersect in vectors such as 10001 and 00101 and 01100. None of these vectors is any closer to any *other* codeword, so we know, receiving one of these, that at least 2 symbol errors have occurred in transmission.

In trying to correct this we'd like to choose the closest codeword, but there is no unique closest. Thus there is really nothing to choose between guessing 00000 and 11101. We pick arbitrarily from these codewords at distance 2 and hope for the best (or perhaps seek retransmission).

0.147. Exercise. Let C be the 3-ary [4,3]-code generated by

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

Find a PCM for C. Hence list the codewords of $C^{\perp}$.

Answer:

First we try to get G in standard form. We subtract two lots of row 1 from row 2:

$$G \mapsto \begin{pmatrix} 1 & 2 & 2 & 1 \\ 0 & -3 & -4 & -1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \equiv \begin{pmatrix} 1 & 2 & 2 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Then subtract two lots of row 3 from row 1; then add row 2 to row 3; then swap rows 2 and 3; then mult row 3 by 2:

$$G \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Now viewing this as $G = [1_3|A]$ we put $H = [-A^t|1_{4-3}]$. Thus

$$H = (-2, 0, -1, 1) \equiv (1, 0, 2, 1)$$

This generates $C^{\perp} = \{0000, 1021, 2012\}$.

0.148. Exercise. Let C be the [3,2] code over $F_4 = \{0, 1, a, b\}$ generated by

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

Explain the meaning of the symbols a and b in this field; write down the addition table for it; and hence or otherwise determine the codewords of C and $C^{\perp}$.

0.149. Exercise. Let C be the binary [7,4] code generated by

$$G = \begin{pmatrix} 1000111 \\ 0100110 \\ 0010101 \\ 0001011 \end{pmatrix}$$

(a) (i) find a PCM H for C

(a) (ii) compute $G.H^t$

(b) show that $d(C) = 3$

(c) (i) show that C is perfect

(c) (ii) how many coset leaders have weight 1?

(d) construct a syndrome look-up table for C

(e) decode the received vector 1110100.

0.150. Exercise. Write down a PCM for the binary Hamming [15,11] code.

Answer:

We need to write down all non-zero vectors in $\mathbb{Z}_2^4$ as columns:

$$H = \begin{pmatrix} 000000011111111 \\ 000111100001111 \\ 011001100110011 \\ 101010101010101 \end{pmatrix}$$