Week 4

Decoding linear codes

Version 2023-10-07. To accessible online version of this chapter

Synopsis. *We explicitly describe a decoder* DECODE: $\mathbb{F}_q^n \to C$ *based on* **coset leaders** *and a* **standard array** *for* C*. For binary* C *sent via a binary symmetric channel, we find the probability* $P_{\text{undetect}}(C)$ *of an undetected transmission error. It is related to the* **weight enumerator** *of* C*. We also find the probability* $P_{\text{corr}}(C)$ *that a codeword is decoded correctly.*

Cosets and coset leaders

It turns out that the following notion is of direct relevance to decoding:

Definition: coset

Given a linear code $C \subseteq \mathbb{F}_q^n$ and a vector $\underline{y} \in \mathbb{F}_q^n$, the **coset** of $\underline{y}$ is the set

$$\underline{y} + C = \{\underline{y} + \underline{c} \mid \underline{c} \in C\}.$$

We recall basic facts about cosets (see for example *Algebraic Structures 1*):

- $C = \underline{0} + C$ is itself a coset. (C is called the *trivial coset*.) Moreover, C is the coset of any codeword $\underline{c} \in C$.
- If $\underline{y}, \underline{z} \in \mathbb{F}_q^n$, then either $\underline{y} + C = \underline{z} + C$ (*if* $\underline{y} - \underline{z} \in C$) or $(\underline{y} + C) \cap (\underline{z} + C) = \varnothing$.
- $\#(\underline{y} + C) = \#C = q^k$.
- There are $\dfrac{\#\mathbb{F}_q^n}{\#C} = q^{n-k}$ distinct cosets.

Thus, the whole space $\mathbb{F}_q^n$ is split (*partitioned*) into q^{n-k} cosets:

$$\mathbb{F}_q^n = C \sqcup (\underline{a}_1 + C) \sqcup \ldots \sqcup (\underline{a}_{q^{n-k}-1} + C).$$

The above is true for any abelian group; but the following is specific to Coding Theory:

Definition: coset leader

A **coset leader** of a coset $\underline{y} + C$ is a vector of minimum weight in $\underline{y} + C$.

Remark: warning — a coset leader may not be unique

There may be more than one coset leader in a coset. However, all coset leaders of a given coset are of the same weight.

Proposition 4.1: the formula for a decoder for a linear code

For a linear code $C \subseteq \mathbb{F}_q^n$, any decoder $\texttt{DECODE}\colon \mathbb{F}_q^n \to C$ satisfies:

$$\forall \underline{y} \in \mathbb{F}_q^n \quad \texttt{DECODE}(\underline{y}) = \underline{y} - \underline{e} \text{ where } \underline{e} \text{ is a coset leader of the coset } \underline{y} + C \text{ of } \underline{y}.$$

Proof. Let $\underline{v} = \texttt{DECODE}(\underline{y})$. Then $\underline{v} \in C$. Put $\underline{e} = \underline{y} - \underline{v}$. Since C is a linear code, $-\underline{v} \in C$, so $\underline{e} = \underline{y} + (-\underline{v}) \in \underline{y} + C$. We have proved that $\underline{e}$ must lie in the coset $\underline{y} + C$.

Vector $\underline{y}$ must be decoded to its nearest neighbour in C, i.e., $d(\underline{y}, \underline{v})$ must be minimised. Yet by Lemma 3.1 $d(\underline{y}, \underline{v}) = w(\underline{y} - \underline{v}) = w(\underline{e})$. Hence the decoder must choose $\underline{e}$ so that $w(\underline{e})$ is minimal in the coset $\underline{y} + C$. By definition, $\underline{e}$ must be a coset leader of $\underline{y} + C$. □

Standard array: construction

We now give a method to construct all cosets and to find one coset leader in each coset.

Definition: standard array

A **standard array** for a linear code $C \subseteq \mathbb{F}_q^n$ is a table with the following properties:

- the table has $|C| = q^k$ columns and q^{n-k} rows;
- each row is a coset;
- the leftmost entry in each row is a coset leader of that row;
- the top row is the trivial coset (i.e., C itself);
- each entry is the sum of the leftmost entry in its row and the top of its column;
- the table contains every vector from $\mathbb{F}_q^n$ exactly once.

We explain how to construct a standard array, using the linear code $C = \{0000, 0111, 1011, 1100\} \subseteq \mathbb{F}_2^4$ as an example.

Row 0 of the standard array: lists all *codevectors* (elements of $C = \underline{0} + C$). They must start from $\underline{0}$, but otherwise the order is arbitrary.

$$0000 \qquad 0111 \qquad 1011 \qquad 1100$$

Row 1: out of vectors not yet listed, choose $\underline{a}_1$ of smallest weight — this guarantees that $\underline{a}_1$ will be a coset leader. Fill in Row 1 by adding $\underline{a}_1$ to each codevector in Row 0.
Say, $\underline{a}_1 = 0001$. To list its coset, add it to row 0: e.g., $0001 + 0111 = 0110$, etc.

$$0001 \qquad 0110 \qquad 1010 \qquad 1101$$

Row 2: choose $\underline{a}_2$ of smallest weight not yet listed, and do the same as for Row 1.
Say, $\underline{a}_2 = 0010$, add it to row 0:

$$0010 \qquad 0101 \qquad 1001 \qquad 1110$$

Row 3: same with, say, $\underline{a}_3 = 0100$:

$$0100 \qquad 0011 \qquad 1111 \qquad 1000$$

Since we have filled 4 rows, and $q^{n-k} = 2^{4-2} = 4$, our standard array is complete:

Example: a standard array for the code $C = \{0000, 0111, 1011, 1100\}$

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 0100 & 0011 & 1111 & 1000 \end{array}$$

Standard array: decoding

Let $C \subseteq \mathbb{F}_q^n$ be a linear code. By Proposition 4.1, any decoder is given by

$$\texttt{DECODE}(\underline{y}) = \underline{y} - \texttt{COSET LEADER}(\underline{y} + C).$$

This suggests the following decoding algorithm for C.

Algorithm 4.2: the standard array decoder

Preparation: construct a standard array for C.
Decoding:

- Receive a vector $\underline{y} \in \mathbb{F}_q^n$.
- Look up $\underline{y}$ in the standard array.
- Return the topmost vector of the column of $\underline{y}$ as $\texttt{DECODE}(\underline{y})$.

Justification: the algorithm is correct because, by definition of a standard array,

(a) Look-up of $\underline{y}$ will succeed as every vector in $\mathbb{F}_q^n$ is present in the array;
(b) the row of $\underline{y}$ starts with $\texttt{COSET LEADER}(\underline{y} + C)$, so
(c) the top of $\underline{y}$'s column is $\underline{y} - \texttt{COSET LEADER}(\underline{y} + C)$ so this is $\underline{y}$ decoded.

Example: use the standard array decoder

For the code $C = \{0000, 0111, 1011, 1100\}$ and standard array constructed above,

- decode the received vectors 0011 and 1100;
- give an example of one bit error occurring in a codeword and being corrected;
- give an example of one bit error occurring in a codeword and not being corrected.

Solution. We work with the following standard array for C:

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 0100 & 0011 & 1111 & 1000 \end{array}$$

The received vector 0011 is in the second column, so $\texttt{DECODE}(0011) = 0111$. The received vector 1100 is a codeword (in the fourth column), so $\texttt{DECODE}(1100) = 1100$.

Suppose that the codeword 0000 is sent. If an error occurs in the last bit, the word 0001 is received and decoded correctly as 0000. If an error occurs in the first bit, the word 1000 is received and decoded incorrectly as 1100.

Discussion: is a standard array decoder unique?

Recall that there may be more than one possible standard array for the code C. Indeed, in the above example the coset $0100 + C$ has two coset leaders: 0100 and 1000. Thus, we could construct a different standard array for C:

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 1000 & 1111 & 0011 & 0100 \end{array}$$

The decoder associated to this standard array is different from the decoder considered above. Both decoders decode the same linear code C. A linear code can have more than one decoder.

However, if C is a **perfect** linear code, then each coset has only one coset leader, so the **decoder is unique.** This property of perfect codes appears on the example sheets.

Reminder (the number of errors corrected by a code)

Recall that a code with minimum distance d corrects $t = [(d-1)/2]$ errors.

The code C in the above example is linear, hence $d(C) = w(C) = 2$ (it is easy to find the minimum weight of the code by inspection). This means that the code corrects $\left[\frac{2-1}{2}\right] = 0$ errors. That is, C is not guaranteed to correct even a single bit error occurring in a codevector. And indeed, we saw in an example how one bit error occurred in a codevector and was not corrected.

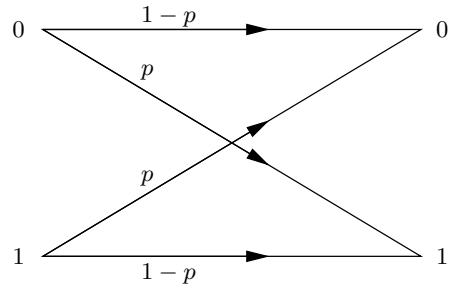
So, from the point of view of Hamming's theory, this code C has no error-correcting capability. It still detects up to one error.

But in Shannon's theory, error-detecting and error-correcting performance of a code are measured probabilistically.

Error-detecting and error-correcting performance of a linear code: Shannon's theory point of view

Shannon's Information Theory is interested in how likely is it that a transmission error in a codeword is not detected/corrected by a decoder of C. We will answer these questions for a binary linear code C transmitted via $BSC(p)$.

Recall that this means that one bit (0 or 1), transmitted via the channel, arrives unchanged with probability $1-p$, and gets flipped with probability p:



When a codeword $\underline{v}$ is transmitted, the channel generates a random error vector and adds it to $\underline{v}$. By definition of $BSC(p)$, for a given $\underline{e} \in \mathbb{F}_2^n$ one has

$$P(\text{the error vector equals } \underline{e}) = (1-p)^{n-i}p^i, \qquad \text{where } i = w(\underline{e}).$$

In determining $P_{\text{undetect}}(C)$, the following notion is very useful:

Definition: the weight enumerator

The **weight enumerator** of a linear code $C \subseteq \mathbb{F}_q^n$ is the polynomial

$$W_C(x,y) = \sum_{\underline{v} \in C} x^{n-w(\underline{v})} y^{w(\underline{v})} = A_0 x^n + A_1 x^{n-1} y + A_2 x^{n-2} y^2 + \ldots + A_n y^n$$

in two variables x, y, where $A_i = \#\{\underline{v} \in C : w(\underline{v}) = i\}$.

Theorem 4.3: $P_{\text{undetect}}(C)$, the probability of an undetected error

Suppose that a codevector of a binary linear code C of length n is transmitted via $BSC(p)$. The probability of an undetected error is

$$P_{\text{undetect}}(C) = W_C(1-p, p) - (1-p)^n.$$

Proof. Let $\underline{v} \in C$ be the codevector being transmitted. Recall that an *undetected error* means that the received vector $\underline{v} + \underline{e}$ is a codevector not equal to $\underline{v}$. Note that, since $\underline{v} \in C$ and C is a vector space,

$$\underline{v} + \underline{e} \in C,\ \underline{v} + \underline{e} \neq \underline{v} \quad \Longleftrightarrow \quad \underline{e} \in C,\ \underline{e} \neq \underline{0}.$$

Therefore, *an undetected error means that the error vector is a non-zero codevector.* We can now calculate

$$P_{\text{undetect}}(C) = \sum_{\underline{e} \in C,\, \underline{e} \neq \underline{0}} P(\text{the error vector is } \underline{e}) = \sum_{\underline{e} \in C,\, \underline{e} \neq \underline{0}} (1-p)^{n-w(\underline{e})} p^{w(\underline{e})}.$$

This is $W_C(1-p, p)$ without exactly one term, excluded by the constraint $\underline{e} \neq \underline{0}$, namely

$$(1-p)^{n-w(\underline{0})} p^{w(\underline{0})} = (1-p)^n,$$

which gives the expression for $P_{\text{undetect}}(C)$ as stated. □

Remark: In general, $P_{\text{undetect}}(C)$ is calculated assuming that the codeword $\underline{v}$ is picked at random from the code. However, our proof shows that for a *linear* binary code and for the binary *symmetric* channel the probability is the same for all codevectors.

Example: calculating the weight enumerator and P_{undetect}

The binary linear code $C = \{0000, 0111, 1011, 1100\}$ has one codeword of weight 0, zero codewords of weight 1, one codeword of weight 2 and two codewords of weight 3. Hence the weight enumerator of C is

$$W_C(x,y) = x^4 + x^2 y^2 + 2xy^3.$$

If a codeword of C is sent via $BSC(p)$, an undetected error occurs with probability

$$P_{\text{undetect}}(C) = (1-p)^2p^2 + 2(1-p)p^3.$$

Discussion. Knowing $P_{\text{undetect}}(C)$ is useful when a code is used for error detection, e.g., if the receiver can request retransmission if an error is detected. Then $P_{\text{undetect}}(C)$ is on average the proportion of incorrect codevectors, hence incorrect symbols, accepted by the receiver. A code should be designed for a particular channel so as to keep this probability below an agreed threshold.

The probability of correct decoding

We will now find the probability of an error being *corrected* for C.

Theorem 4.4: $P_{\text{corr}}(C)$, the probability of correct decoding

Suppose that a codevector of a binary linear code C is transmitted via $BSC(p)$. The probability that the received vector will be decoded correctly is

$$P_{\text{corr}}(C) = \sum_{i=0}^{n} \alpha_i (1-p)^{n-i}p^i,$$

where α_i denotes the number of cosets where the coset leader is of weight i.

Proof. Recall that $\underline{v} \in C$ is *decoded correctly* if $\texttt{DECODE}(\underline{v}+\underline{e}) = \underline{v}$. By Proposition 4.1,

$$\begin{aligned}\texttt{DECODE}(\underline{v}+\underline{e}) &= \underline{v}+\underline{e}-\texttt{COSET LEADER}(\underline{v}+\underline{e})\\ &= \underline{v}+\underline{e}-\texttt{COSET LEADER}(\underline{e}).\end{aligned}$$

Therefore, *correct decoding occurs if the error vector is the chosen coset leader of its coset*.

We therefore have one good outcome per coset: namely, $\underline{e}$ equals the chosen coset leader of the coset. Recall that that happens with probability $(1-p)^{n-i}p^i$ where i is the weight of the coset leader of the given coset. Summing over all cosets and gathering the like terms, we obtain the formula for $P_{\text{corr}}(C)$ as stated (it does not depend on $\underline{v}$). □

Discussion. When is it important to know $P_{\text{corr}}(C)$? In one-way communication channels without retransmission even if an error is detected, the decoder produces a best guess as to which codevector was sent. An example is computer memory, where information could have been written ("sent") long time ago, and it is not possible to "resend" it. Thus, $1-P_{\text{corr}}(C)$ is on average the proportion of incorrect codevectors, hence incorrect symbols, accepted by the receiver. A code should be designed for a particular channel to keep $1-P_{\text{corr}}(C)$ below an agreed threshold.

Example: calculation of P_{corr}

Let $C = \{0000, 0111, 1011, 1100\}$. From the standard array

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 1000 & 1111 & 0011 & 0100 \end{array}$$

we can see that $\alpha_0 = 1$, $\alpha_1 = 3$, $\alpha_2 = \alpha_3 = 0$ (given by the leftmost column), so

$$P_{\text{corr}}(C) = (1-p)^4 + 3(1-p)^3 p.$$

Comparing codes using approximate values of P_{undetect} or P_{corr}

Our analysis above gives, for a binary linear code C transmitted via $BSC(p)$, the probabilities $P_{\text{undetect}}(C)$ and $P_{\text{corr}}(C)$ as polynomials in p.

In practical situations p is typically very small, so it is rarely useful to know all terms of these polynomials in p. The term with the lowest power of p will dominate and can be used as an approximate value of the probability. This makes comparing codes easier.

Example. We compare use of the code C against transmission of undencoded information ("trivial binary code of length 1").

If C is used for error detection: $P_{\text{undetect}}(C) = (1-p)^2 p^2 + 2(1-p)p^3$. This is a polynomial of the form $p^2 + o(p^2)$ where $o(p^2)$ contains powers of p higher than 2. For small p, the terms in $o(p^2)$ are negligible compared to p^2. We conclude:

$$P_{\text{undetect}}(C) \sim p^2 \text{ whereas } P_{\text{undetect}}(\text{no encoding}) = p,$$

i.e., the use of C improves the proportion of bad bits in the output from p to p^2.

If C is used for error correction: $1 - P_{\text{corr}}(C) = 1 - (1-p)^4 - 3(1-p)^3 p = p + o(p)$ (check this by opening the brackets). Thus,

$$1 - P_{\text{corr}}(C) \sim p \text{ whereas } 1 - P_{\text{corr}}(\text{no encoding}) = p.$$

Hence C is useless for error correction: its use does not improve the proportion of bad bits in the output while increasing the volume of information transmitted two-fold ($R = 0.5$).

The above suggests that for error correction, more mathematically sophisticated codes need to be designed. In the rest of the course, we will see how ideas from different areas of mathematics are used in code constructions.