MATH32031: Coding Theory • Part 8: Decoding Linear Codes (II)

Now we analyze the decoding algorithm we have just described. Remember that we still have the usual algorithm for detecting an error: if the received vector is not a codeword then there is an error.

Proposition 13

(a) *A received vector is decoded correctly if and only if the error vector is a coset leader.*

(b) *The algorithm fails to detect that an error occurred if and only if the error vector is in C but is not $\underline{0}$.*

(c) *This is a nearest neighbour decoding.*

Proof. (a) If $\underline{x}$ is the sent codeword and $\underline{e}$ is the error then the received vector is $\underline{y} = \underline{x} + \underline{e}$. If $\underline{y} \in \underline{a}_i + C$ then we decode it as $\underline{y} - \underline{a}_i$, which is equal to $\underline{x} + \underline{e} - \underline{a}_i$. This is equal to the sent vector $\underline{x}$ if and only if $\underline{e} = \underline{a}_i$.

In particular, if we have decoded correctly then $\underline{e}$ is a coset leader. Conversely, if $\underline{e}$ is a coset leader then

$$\underline{e} \in \underline{e} + C = \underline{e} + \underline{x} + C = \underline{y} + C = \underline{a}_i + C,$$

so being coset leader forces $\underline{e} = \underline{a}_i$.

(b) $\underline{e} = \underline{y} - \underline{x}$ and $\underline{x} \in C$, so $\underline{y} \in C$ if and only if $\underline{e} \in C$. Thus we declare that there is an error if and only if $\underline{e} \notin C$. We miss an error if and only if $\underline{e} \neq \underline{0}$ (so there is actually an error) and $\underline{e} \in C$.

(c) Write $\underline{y} = \underline{a}_i + \underline{c}$. We want to show that $d(\underline{y}, \underline{c}) \leq d(\underline{y}, \underline{c}')$ for any $\underline{c}' \in C$. This amounts to proving that $w(\underline{a}_i) = w(\underline{y} - \underline{c}) \leq w(\underline{y} - \underline{c}')$. But notice that

$$\underline{y} - \underline{c}' \in \underline{y} + C = \underline{a}_i + C,$$

so, by definition of coset leader, $w(\underline{a}_i) \leq w(\underline{y} - \underline{c}')$. □

Remark. In practice this decoding algorithm is too slow for large codes; indeed, the standard arrays for many large codes have never been worked out. This will lead us later to the technique of "syndrome decoding".

Example. From previous work we know a code C with a standard array:

0000	1011	0101	1110
1000	0011	1101	0110
0100	1111	0001	1010
0010	1001	0111	1100

.

Note the following 1-error "trajectories"

$$\begin{array}{ccccc} \text{sent} & & \text{received} & & \text{decoded} \\ 0101 & \longrightarrow & 0001 & \longrightarrow & 0101 \\ 0101 & \longrightarrow & 0100 & \longrightarrow & 0000 \end{array}$$

Thus we see that some 1-error messages are not decoded as the original message. (This is, of course, only to be expected since the minimum distance of the code is 2, thus $t = 0$).

This leads us to calculate the probability of correctly decoding (via this algorithm) to get the original message sent.

Definition. Let $P_{\text{corr}}(C)$ denote the probability that a received vector is correctly decoded. Let α_i denote the number of coset leaders of weight i.

Theorem 14 *Let C be an $[n, k]$ $\mathbf{F}_2$-code transmitted by a binary symmetric channel with symbol error rate r. Then*

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

Proof. From the Proposition we see that

$$P_{\text{corr}}(C) = \text{Prob(error } \underline{e} \text{ is a coset leader)} = \sum_{i=0}^{n} \text{Prob(}\underline{e} \text{ is a coset leader of weight } i).$$

But we have seen in earlier lectures that

$$\text{Prob(}\underline{e} \text{ is a given vector of weight } i) = r^i(1-r)^{n-i},$$

so

$$\text{Prob(}\underline{e} \text{ is a coset leader of weight } i) = \alpha_i r^i (1-r)^{n-i}$$

and thus

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

$\square$

Example (again). We compute $P_{\text{corr}}(C)$ for the code C discussed above. By inspection $\alpha_0 = 1$, $\alpha_1 = 3$, $\alpha_i = 0$ for $i > 1$. So

$$\begin{aligned} P_{\text{corr}}(C) &= (1-r)^4 + 3r(1-r)^3 \\ &= (1-r)^3(1-r+3r) \\ &= (1-r)^3(1+2r). \end{aligned}$$

For instance: if $r = .01$ then $P_{\text{corr}}(C) = .9897$, i.e. $P_{\text{err}}(C) = .0103$.

On the other hand, if a 2-digit message is sent, without any coding, via the same channel, then

$$P_{\text{corr}}(C) = (1-r)^2,$$

which would give

$$P_{\text{err}}(C) = .0199.$$

So we have essentially halved the error probability at the cost of sending 2 check symbols per message.

Definition. $P_{\text{undetect}}(C)$ is the probability that the received vector contains an error that we do not detect.

We have seen in the Proposition that there is an undetected error if and only if $\underline{e} \in C$ and $\underline{e} \neq \underline{0}$.

Let A_i be the number of codewords of weight i. Then

$$\begin{aligned} P_{\text{undetect}}(C) &= \sum_{\underline{c} \in C \setminus \{\underline{0}\}} \text{Prob}(\underline{e} = \underline{c}) \\ &= \sum_{i=1}^{n} A_i r^i (1-r)^{n-i}. \end{aligned}$$

Example (yet again). By inspection $A_1 = 0$, $A_2 = 1$, $A_3 = 2$. So

$$\begin{aligned} P_{\text{undetect}}(C) &= (1-r)^2 r^2 + 2r^3(1-r) \\ &= r^2(1-r^2). \end{aligned}$$

In particular, setting $r = .01$, we find that

$$P_{\text{undetect}}(C) = .000099\ldots,$$

which is not bad!