A Note on Hamming Code

Szabolcs Mikulás

The Hamming code is a powerful error correcting code. It enables us to detect errors and to recover the original binary word if one digit goes wrong (even if we do not know which bit went wrong).

We will use that every non-zero natural number can be written as a binary number, i.e., $n = \sum\{2^{k_i} : i \in I\}$ for some index set I. Note that the exponents $\{k_i : i \in I\}$ are all different and uniquely determined. Thus we can define a function f from positive natural numbers into the subsets of natural numbers such that

$$n = \sum\{2^i : i \in f(n)\}.$$

For example, $f(1) = \{0\}$, $f(2) = \{1\}$, $f(3) = \{0, 1\}$, etc.

Conversely, we can define a function g that associates to every power of 2, 2^k, those positive natural numbers n such that 2^k occurs in the sum for n:

$$g(2^k) = \{n : k \in f(n)\}.$$

For instance, $g(2^0) = \{1, 3, 5, 7, 9, 11, \ldots\}$, $g(2^1) = \{2, 3, 6, 7, 10, \ldots\}$, $g(2^2) = \{4, 5, 6, 7, 12, \ldots\}$ etc.

Let $s = (s_1, s_2, \ldots, s_{2^n})$ be a 2^n-long binary word (i.e., every s_j is either 0 or 1). The *Hamming code*, $H(s)$, of s is a $(2^n + n + 1)$-long word defined as follows: for $1 \leq i \leq 2^n + n + 1$,

$$H(s)_i = \begin{cases} \sum\{s_{j-1-\lfloor \log_2 j \rfloor} : j \in g(i)\} \pmod 2 & \text{if } i = 2^k \text{ for some } 0 \leq k \leq n \\ s_{i-1-\lfloor \log_2 i \rfloor} & \text{otherwise.} \end{cases}$$

For example, if $s = (1, 0, 0, 0)$, then $H(s) = (1, 1, 1, 0, 0, 0, 0)$. Indeed,

$$\begin{aligned} H(s)_1 &= H(s)_3 + H(s)_5 + H(s)_7 \pmod 2 = s_1 + s_2 + s_4 \pmod 2 = 1 \\ H(s)_2 &= H(s)_3 + H(s)_6 + H(s)_7 \pmod 2 = s_1 + s_3 + s_4 \pmod 2 = 1 \\ H(s)_4 &= H(s)_5 + H(s)_6 + H(s)_7 \pmod 2 = s_2 + s_3 + s_4 \pmod 2 = 0. \end{aligned}$$

In general, we will refer to the bits coming from the original binary word s as data bits, and the rest as parity bits. In the above example, the parity bits are the first two occurrences of 1 and the first occurrence of 0.

Let n be fixed and t be a $(2^n + n + 1)$-long binary word such that

- there is no 2^n-long binary word s for wich $t = H(s)$,
- if we change a certain bit in t, then it becomes the Hamming code for some 2^n-long binary word u.

We claim that u can be recovered from t.

By assumption there is precisely one incorrect bit in t, but we do not know which one. Consider the following algorithm. Let the sequence v consist of the data bits of t and its Hamming code be $H(v)$. There are two cases.

Case 1: one data bit t_i $(i \neq 2^k)$ is incorrect. Then the parity bits will be different in t and in $H(v)$ for precisely those indices j such that $j \in f(i)$. By the definition of the function f, the sum of these indices yields i. Thus we know which data bit t_i is incorrect. Note also that there are more than one parity bits in t and $H(v)$ which disagree.

Case 2: one parity bit t_i $(i = 2^k)$ is incorrect. Then all the other parity bits are correct. Thus changing t_i in t yields a binary word such that it is the Hamming code of v.

Finally note that cases 1 and 2 can be distinguished by the number of parity bits that differ in t and in $H(v)$. Thus we know which one of the cases apply.

As an example let us look at the binary word $t = (1, 1, 1, 0, 0, 1, 0)$. Then $v = (1, 0, 1, 0)$ and $H(v) = (1, 0, 1, 1, 0, 1, 0)$. Hence the disagreeing parity bits are $t_2 = 1 \neq 0 = H(v)_2$ and $t_4 = 0 \neq 1 = H(v)_4$. Thus case 1 above applies and we conclude that we should change $t_{2+4} = t_6$. Indeed, if you take the sequence $t' = (1, 1, 1, 0, 0, 0, 0)$, it turns out to be the Hamming code of $(1, 0, 0, 0)$.

Now consider the binary word $t = (1, 0, 1, 0, 0, 0, 0)$. Then $v = (1, 0, 0, 0)$ and $H(v) = (1, 1, 1, 0, 0, 0, 0)$. Thus case 2 applies, and we get the correct Hamming code by changing the second bit in t from 0 to 1.