Note for lecture 4

The error correction/detection techniques covered here are based on the following book:

- Joseph A. Gallian. Contemporary Abstract Algebra. 7th edition. Brooks/Cole, 2010. ISBN 9780495831532 Chapter 31.

Note that in the following, we use the same definition of a "code word" as defined in lecture 3.
Same applies for the definition of a "linear code" (i.e., a parity check code)

Definition 1. Hamming distance between two code words is the # of different bits between the two and is denoted by $d(u,v)$ for code words u and v. (This is essentially the minimum distance we defined in lecture 3)

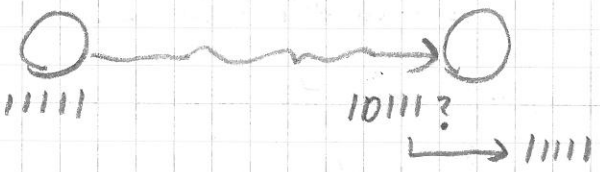
Definition 2. Hamming weight of a code word is the # of non-zero bits in it, denoted by $wt(u)$ for code word u.
Hamming weight for a linear code is the minimum Hamming weight of any non-zero code word in the linear code.

Example: let $u=\{00011\}$, $v=\{00010\}$, $w=\{00000\}$
then $d(u,v)=1$, $d(u,w)=2$, $wt(u)=2$
The Hamming weight of $\{u,v,w\}$ is 1.

Definition 3. The nearest-neighbor rule.

In this rule, error correction is performed by converting the received code word into the code word that has the smallest Hamming distance to the received code word.

Example: Suppose there are two code words, 11111 and 11001, and that data sender sent 11111 but data receiver got 10111 due to some channel distortion. Using the nearest-neighbor rule, the data receiver can convert 10111 to 11111 since $d(11111,10111)=1$ and $d(11001,10111)=3$, and that corrects the error.



Question: is there any performance guarantee of using the nearest-neighbor rule to correct errors?

Answer: Yes see ↓

Theorem 1. $d(u,v) = wt(u-v)$ for code words u and v.

Proof idea: In modulo-2 substraction of u by v, the result is a code word having 1s for the bits where u and v differs and 0s for the bits where u and v agrees.

Theorem 2. For any code words u, v, and w,

$$d(u,v) \le d(u,w) + d(w,v).$$

Proof idea:

Theorem 3 (main result!). If the Hamming weight of a linear code is at least $2t+1$, then the nearest-neighbor rule can correct any t or fewer errors; alternatively, it can detect any $2t$ or fewer errors.

Proof idea: Suppose the original code word is u, and the received version is v, and w is any code word other than u.

Then since

$$\begin{aligned} 2t+1 &\le wt(w-u) = d(w,u) \\ &\le d(w,v) + d(v,u) \\ &\le d(w,v) + t \end{aligned}$$

which implies $d(w,v) \ge t+1$.

By definition we have $d(u,v) \le t$.

Therefore u is the closest code word to v, and thus using the nearest-neighbor rule in this case we can successfully conect the error.

Geometric illustration: