Note for lecture 17

The error correction/detection techniques covered today are based on this textbook:

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

The automotive data communication introduced today includes the CAN specification V2, part B, 1991 ROBERT BOSCH GmbH.

Definition 1. Hamming distance between two codes (i.e., bit vectors) is the # of different bits between the two and is denoted by $d(u,v)$ for codes u and v.

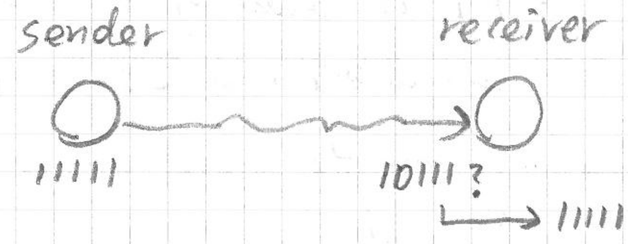
Definition 2. Hamming weight of a code is the # of non-zero bits in the code, denoted by $wt(u)$ for code u.
Hamming weight for a set of codes is the minimum Hamming weight among all non-zero codes in the set.

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 into the code word that has the smallest Hamming distance to the received code.

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 ↓

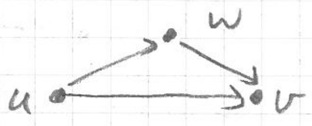
Theorem 1. $d(u,v) = wt(u-v)$ for codes u and v.

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

Theorem 2. For any codes 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 set of codes 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

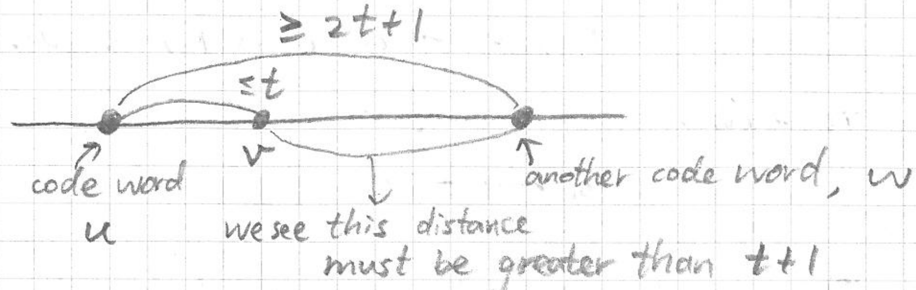
$$\begin{aligned} 2t+1 &\le wt(w-u) = d(w,u) \\ &\le d(w,v) + d(v,u) \\ &\le d(w,v) + t \end{aligned}$$

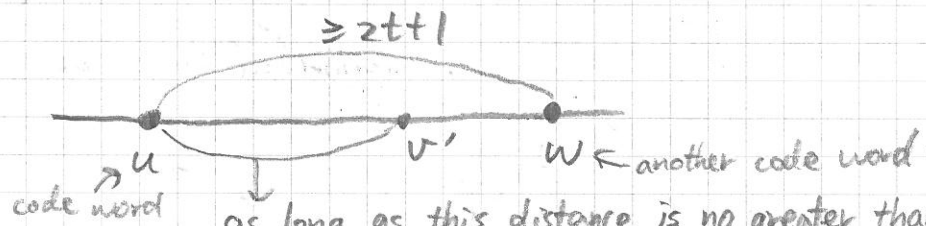
which implies $d(w,v) \ge t+1$.

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 correct the error.

Geometric illustration:





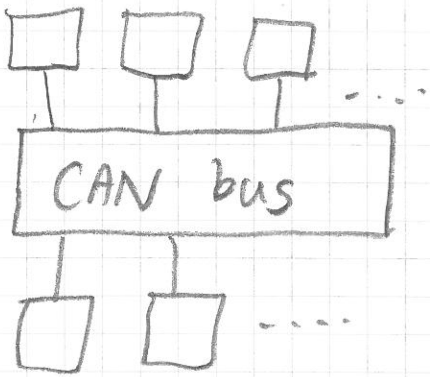
as long as this distance is no greater than $2t$, we can assure that the received code v' can never be mistakenly identified as a code word; in other word, the error can always be detected (since it is not a code word).

- CAN bus in automotive data communication:

Modern vehicles heavily rely on computing chips (known as ECU, electronic control unit) to monitor and coordinate in-vehicle subsystems such as airbags, safety driving features, wheels, and infotainment.

CAN (Controller Area Network) is designed to enable efficient and robust data communications between these chips.

The CAN bus consists of a single channel that carries bits. Chips connected to a bus can send and receive bits, and they can also monitor the current bit value on the bus.



Each data sent on a bus is considered as a broadcast.

(compare it to the ALOHA protol and review how ALOHA resolves data collisions).

Different from the ALOHA protocol, the CAN bus features an arbitration process that can

1. waste no delay in allowing one of the data to be sent successfully;
2. prioritize data communication.

The priority is encoded in the IDENTIFIER field of a data frame. The field is a string of logical values "dominant" an "recessive". In case of a wired-OR implementation of the bus, the "dominant" level would be represented by a logical "1" and the "recessive" level by a logical "0." Hence, the dominant bit will prevail the recessive bit when concurrently sent on the bus. During arbitration, every data sender compares the level of the bit sent with the level that is monitored on the bus. If both levels are equal, the sender can continue to send; if a "recessive" level is sent but a "dominant" level is monitored, the sender must then withdraw and wait until the next frame. Arbitration starts at MSB.

Example:

	MSB … LSB
d_1	0001110010 1
d_2	0011110001 0
d_3	0010011010 1

$\longrightarrow$

then the relative priority: $d_2 > d_3 > d_1$