Lecture of January 21th 2004
Scribe: Michael Wood

1 Error Detection (continued)

Hamming Distance

Recall from lecture 1 that Hamming Distance is defined as: $d_H(\underline{x}, \underline{y}) = \sum_{i=1}^{n} \delta[x_i \neq y_i]$. Also recall that the minimum Hamming distance of a code is defined as: $d = min\{d_H(\underline{c_i}, \underline{c_j}) : \underline{c_i} \in C \wedge \underline{c_j} \in C \wedge \underline{c_i} \neq \underline{c_j}\}$.

Another useful value is Hamming weight $w_H(\cdot)$ which is the Hamming distance of a codeword from the zero codeword. Formally:

$0 \leq w_H(\underline{c}) = d_H(\underline{c}, \underline{o}) \leq n$ where $\underline{c} \in C$ is an n-tuple vector and $\underline{o} = \{0\}^n$.

The error detecting capability of a code C is expressed in terms of Hamming weights and distances. In particular, it takes at least d changes to change one codeword into another, where d is the minimum Hamming distance of the code thus every error pattern of $w_H \leq d-1$ is detectable.

Example:

$C = \{000, 011, 101, 110\} \Rightarrow d = 2$
$\Rightarrow$ any error pattern of $w_H \leq 1$ is detectable
$\Rightarrow$ any single error is detectable.

Note however that the error detecting capability of a code is a lower bound, meaning that we are guaranteed to detect errors if the error pattern has $w_H \leq d-1$; however, we may still detect error patterns of $w_H > d-1$.

$C = \{000, 011, 101, 110\} \Rightarrow d = 2 \Rightarrow$ single error detection.

$000 \rightarrow \oplus \rightarrow 111$ (error has occurred and is detectable even though the number of errors $> d-1$)
$\uparrow$
111 (error pattern)

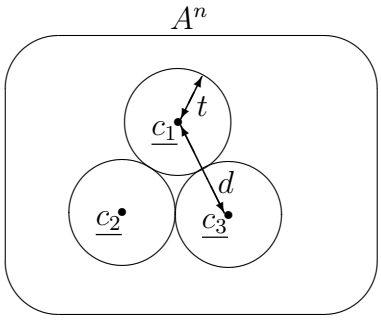
(Here the result is 'bitwise' mod 2 addition of the input and the error pattern.)

2 Error Correction

Imagine the observation space A^n of all the possible received vectors. It would be common sense to decode any $\underline{r} \in A^n$, but $\underline{r} \notin C$ to the closest codeword to it in C (in terms of Hamming distance) because that is clearly the most probable codeword to have been sent. As codewords are separated by at least d locations, any number of errors less than $\frac{d}{2}$ must be correctable.

The error correcting capability of code (denoted by t) is expressed in terms of Hamming weights and distances. Where d is the minimum Hamming distance of the code, every error pattern of $w_H \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ is correctable. Hence $t = \left\lfloor \frac{d-1}{2} \right\rfloor$.

Both error detection and correction capabilities can be visualized in the following manner. The set of all possible words $\underline{w} \in A^n$ is represented by the rounded rectangle in the following figure. Some of the words are codewords, marked as points $\underline{c_i}$. Circles are drawn around the codewords at radius $\left\lfloor \frac{d-1}{2} \right\rfloor$. It should be clear that if less than or equal to $d-1$ errors have occurred in transmission of codeword $\underline{c_1}$ then it is not possible for the result to be any other $\underline{c_i} \in C$; Furthermore, if less than or equal to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors have occurred in transmission of codeword $\underline{c_1}$ then it must still be within the radius of the circle about $\underline{c_1}$ and therefore is closest to $\underline{c_1}$ than any other $\underline{c_i}$. Hence by the closest codeword scheme, we can definitely correct the error. Note that it is possible for an arbitrarily large number of errors to occur in transmission, thereby allowing the received value to be any $\underline{w} \in A^n$. If this occurred, then our closest codeword correction scheme would be in error, but the probability of such an event is usually vanishingly small.



Hamming distance is a metric.

A metric is a non-negative symmetric value which enjoys the triangle property.

The triangle property can be thought of as the behavior of "real" distances. For example, euclidean distance enjoys the triangle property. It is formally defined below by three requirements.
$\forall \underline{x}, \underline{y}, \underline{z} \in X, \quad d(\cdot) : X \times X \to \mathbb{R}$

1. $d(\underline{x}, \underline{y}) \geq 0$ (equivalent only when $\underline{x} = \underline{y}$)
2. $d(\underline{x}, \underline{y}) = d(\underline{y}, \underline{x})$
3. $d(\underline{x}, \underline{y}) + d(\underline{x}, \underline{z}) \geq d(\underline{y}, \underline{z})$

$d(\cdot)$ enjoys the triangle property only if 1 and 2 and 3 are true.

Lemma 1 $\forall \underline{x}, \underline{y}, \underline{z} \in X : \delta[\underline{x} \neq \underline{y}] + \delta[\underline{x} \neq \underline{z}] \geq \delta[\underline{y} \neq \underline{z}]$

Proof: (by contradiction)

Recall: $\delta[true] = 1$, $\delta[false] = 0$

Assume: $\delta[\underline{x} \neq \underline{y}] + \delta[\underline{x} \neq \underline{z}] < \delta[\underline{y} \neq \underline{z}]$

Case 1: $\delta[\underline{y} \neq \underline{z}] = 1$
$\Rightarrow \delta[\underline{x} \neq \underline{y}] + \delta[\underline{x} \neq \underline{z}] = 0$
$\Rightarrow (\underline{x} = \underline{y}) \wedge (\underline{x} = \underline{z}) \Rightarrow \underline{y} = \underline{z}$
$\Rightarrow \delta[\underline{y} \neq \underline{z}] = 0 \Rightarrow$ *contradiction.*

Case 2: $\delta[\underline{y} \neq \underline{z}] = 0$
$\Rightarrow \delta[\underline{x} \neq \underline{y}] + \delta[\underline{x} \neq \underline{z}] < 0$
$\Rightarrow$ *contradiction. (because* $d(\cdot)$ *is non-negative)*

∎

⋆ **Theorem:** Hamming Distance is a metric.

Proof:
1. Hamming Distance is non-negative by definition.
2. Hamming Distance is symmetric by definition.
3. Hamming Distance enjoys the triangle property:

$$\begin{aligned} d(\underline{x},\underline{z}) + d(\underline{x},\underline{y}) &= \textstyle\sum_i \ \delta[\underline{x_i} \neq \underline{z_i}] + \sum_i \ \delta[\underline{x_i} \neq \underline{y_i}] \\ &= \textstyle\sum_i \ (\delta[\underline{x_i} \neq \underline{z_i}] + \delta[\underline{x_i} \neq \underline{y_i}]) \\ &\geq \textstyle\sum_i \ \delta[\underline{z_i} \neq \underline{y_i}] (by\ Lemma 1) \\ &\geq d(\underline{y},\underline{z}) \end{aligned}$$

∎

We can now use this result to prove our earlier claims about error correction.

⋆ **Theorem:** Let C be a code with minimum Hamming distance d using the "closest codeword" decoding rule. Then C can correct all error patterns of Hamming weight $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$

Proof:
Transmit $\underline{c} \in C$ and receive $\underline{r} \in A^n$
Suppose e errors happen in transmission; therefore, $d(\underline{c},\underline{r}) = e$.
$\forall \underline{w} \in C : \underline{w} \neq \underline{c}, \quad d(\underline{w},\underline{c}) \geq d$ by definition.
Since $\underline{r}$ is at most e away from $\underline{c}$ it can travel at most e from $\underline{c}$ to $\underline{w}$.
Therefore, $d(\underline{w},\underline{r}) \geq d - e$.

Now, e is only correctable if $d(\underline{w},\underline{r}) > d(\underline{c},\underline{r})$
Therefore, e is correctable if $d(\underline{w},\underline{r}) > e$

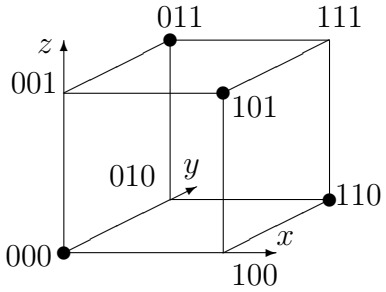
Therefore, as long as $d - e > e$ then e is correctable.
i.e. $e < \frac{d}{2} \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ errors are correctable.
Hence, the error correcting capability of a code is $t \leq \left\lfloor \frac{d-1}{2} \right\rfloor$

∎

A geometrical representation for codewords of a code C of length n can be given by points of a discrete space.

Example:

$A = \{0, 1\}$, $C = \{000, 011, 101, 110\}$
$n = 3$, $|A^n| = 8$, $|C| = 4$, $d = 2$, $t = 0$
In this example we can detect $d - 1 = 1$ error, and cannot correct any errors.



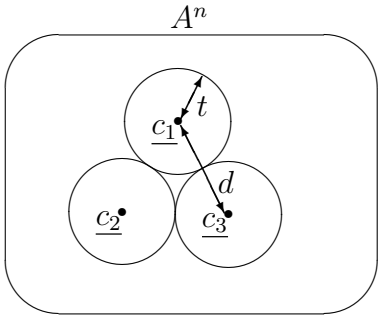
To move in the x direction, flip the 1^{st} coordinate.
To move in the y direction, flip the 2^{nd} coordinate.
To move in the z direction, flip the 3^{rd} coordinate.

Here half of the points are redundant, giving error resilience.

3 Hamming Bound

We have $C \subset A^n$, and we want to maximize both bandwidth efficiency and power efficiency. To maximize bandwidth efficiency, we want to maximize the rate of our code = the number of information bits transmitted in an interval = $R = \frac{log_2|C|}{n}$. Hence we want $|C|$ to be large, i.e. make lots of A^n into codewords, but that decreases d and to maximize power efficiency, we must maximize d.

The Hamming bound is an upper bound on the number of codewords $|C|$ in a t-error-correcting code over alphabet A with length n, where n and t are fixed. To develop the Hamming bound we use the geometrical representation just presented. In order for the code $C \subseteq A^n$ to have an error correction capability of t, spheres of radius t centered at he codewords must be disjoint. This reduces to the discrete sphere packing problem where each codeword is a discrete sphere of radius t in A^n



The number of points or elements in each Hamming sphere of radius t is given by:

$$1 + \binom{n}{1}(|A|-1)^1 + \binom{n}{2}(|A|-1)^2 + ... + \binom{n}{t}(|A|-1)^t$$

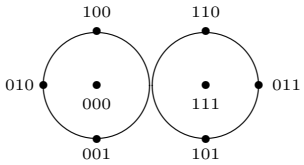
(The $(|A|-1)^t$ term is necessary because it is not necessarily binary.)

The Hamming bound is then given by:

$$|C| \leq \frac{|A|^n}{\sum_{i=0}^{t}\binom{n}{i}(|A|-1)^i}$$

Perfect sphere packing occurs when $\frac{\sum sphere\ volume}{total\ volume} = 1$. Perfect codes are therefore equivalent to perfect discrete sphere packing. Perfect codes satisfy the Hamming bound with equality. Perfect codes do not necessarily achieve capacity in A^n, but only achieve capacity in A^n bounded by the fixed condition of t.

Example:
$C = \{000, 111\}, \quad d = 3, \quad t = 1$
This is a perfect code as illustrated in the figure below. The same is true for any binary repetition code of odd length. A binary repetition code of length n is denoted by R_n.



Other perfect codes:

Hamming Codes ($t = 1$) are prefect.
For any n, the maximum number of codewords in a single error correcting code $= |C| \leq \frac{2^n}{1+n}$

Example
$n = 7, \quad |C| \leq \frac{2^7}{1+7} = 2^4, \quad$ i.e. (7,4,3)

Golay code (23,12,7) has perfect discrete sphere packing. It is a triple error correcting code.

$$|C| \leq \frac{2^{23}}{1 + 23 + \binom{23}{2} + \binom{23}{3}} = 2^{12}$$

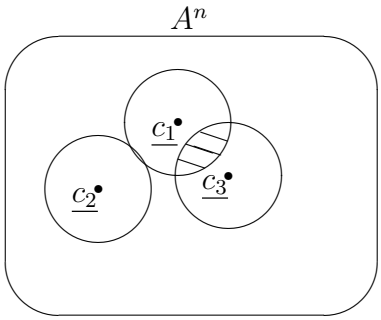
4 Gilbert Bound

The Gilbert bound gives the number of codewords guaranteed in a code with a certain error correcting capability. This is a lower bound on the number of codewords in the optimal code (i.e. the code with the most codewords having a certain minimum distance). We can describe the Gilbert bound algorithmically.

Algorithm:

Fix: $A,\ n,\ t$
Let $C = \emptyset,\ i = 1$
Label all points in A as "non-excluded"
While there are non excluded points, do the following:
  Choose any word $c_i \in A^n$ and add it to C
  i.e. $C \Leftarrow C \cup \{c_i\}$
  mark c_i and any $w \in A^n : d(c_i, w) \leq 2t$ as excluded
  $i \Leftarrow i + 1$
End While Loop

This algorithm generates a code such that the resulting minimum distance d is equal to $2t + 1$. Note that in construction, the Hamming spheres (of radius 2t) are allowed to overlap, as shown in the figure below. This implies that each iteration does not necessarily exclude all words in the sphere of the new codeword, because some may have been previously excluded.



The maximum number of words excluded in each run of the algorithm is given by the following:

$$1 + \binom{n}{1}(|A|-1)^1 + \binom{n}{2}(|A|-1)^2 + ... + \binom{n}{2t}(|A|-1)^{2t}$$

The gilbert bound is then given by:

$$|C| \geq \frac{|A|^n}{\sum_{i=0}^{2t}\binom{n}{i}(|A|-1)^i}$$

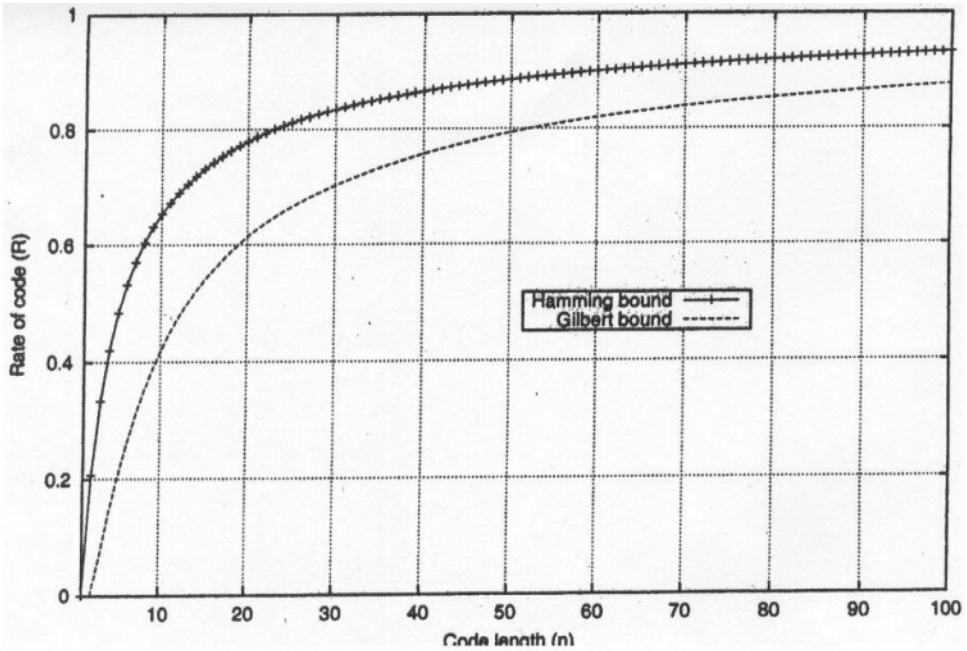
This achieves error correcting capability of exactly t.

Example:
$|A| = 2, \quad t = 1$

$$\frac{2^n}{1 + n + \frac{n(n-1)}{2}} \leq |C| \leq \frac{2^n}{1+n}$$

$$1 - \frac{1}{n}log_2(1 + n + \frac{n(n-1)}{2}) \leq \frac{log_2(|C|)}{n} = R \leq 1 - \frac{1}{n}log_2(1+n)$$

The following figure shows the relation between the Hamming and gilbert bounds. As the bounds get closer our options increase, i.e. more possible codebooks, but those codebooks have increasingly similar behaviors.



5 Decoders

In Hard Decision Decoders (HDD), the demodulator makes hard decisions on the received information (e.g. bits and BSC) and sends the decided symbols/bits to the decoder without passing on any further information on the certainty of its decisions. For example, with +5v for 1, -5v for 0 sent and x received, no information on how close x was to +5v or -5v will be passed on with the decoder decision. In contrast, soft decision decoders (SDD) involve demodulators which somehow pass on some information about the certainty of their decisions. Given a practical decoding algorithm which can make use and sense of the extra reliability data, a soft decision decoder can perform a better job in pinpointing the positions of more likely errors. Hard decision decoders are of two types:

1. Complete: output $\in C$.

Definition: Given code $C \subset A^n$ and any vector $\underline{r} \in A^n$, a complete decoder is one which produces a codeword $\underline{w} \in C$, i.e. it is a function from $A^n toC$

Hence a complete decoder is really partitioning A^n and forcing whatever it receives into one of the partitions. Note that the regions are disjoint and complete, i.e. $R_1 \dot{\cup} R_2 \dot{\cup} ... = A^n$

2. Incomplete: output $\in \{C \cup failure\}$ (i.e. retransmission is possible.)

Definition: An incomplete decoder is <u>not</u> guaranteed to produce a codeword. For $\underline{r} \in A^n$ the decoder may give a "decoding failure".

It is very important to note that incomplete decoders are not inferior to complete decoders.

$$\text{Modulator} \rightarrow \text{Channel} \rightarrow \text{Demodulator} \overset{\text{word}}{\underset{\downarrow}{\rightarrow}} \text{Decoder} \overset{\text{codeword}}{\underset{\text{output}}{\underset{\downarrow}{\rightarrow}}}$$

We may model this as a binary erasure channel (BEC) if using incomplete decoding. To minimize word/block/frame error probability, we showed:

$$\hat{v} = \underset{\underline{v}}{\overset{\text{MAP}}{argmax}} P(\underline{v}|\underline{r}) \rightarrow \underset{\underline{v}: \text{ a priori equiprobable}}{\downarrow} \hat{v} = \underset{\underline{v}}{\overset{\text{ML}}{argmax}} P(\underline{r}|\underline{N})$$

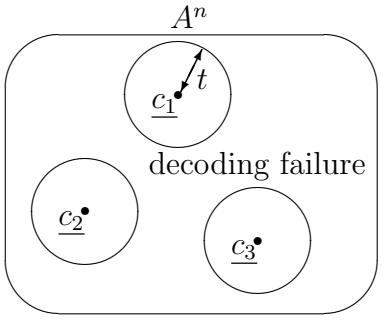
Consider a BSC, crossover probability $= \varepsilon$, want $P(\underline{r}|\underline{v}) \uparrow$

$$\begin{aligned} P(\underline{r}|\underline{v}) &= \varepsilon^{d_H(\underline{r},\underline{v})} \cdot (1-\varepsilon)^{n-d_H(\underline{r},\underline{v})} \quad \text{condition } 0 \leq \varepsilon \leq \tfrac{1}{2} \\ &= (1-\varepsilon)^n (\tfrac{\varepsilon}{1-\varepsilon})^{d_H(\underline{r},\underline{v})} \end{aligned}$$

Therefore, $P(\cdot) \uparrow$ needs $d_H(\underline{r},\underline{v}) \downarrow$
Therefore, MAP is reduced to Minimum Hamming Distance. The partitioning is based on a rule that maximizes the probability of correct decision by minimizing the Hamming distortion (Hamming distance).

Definition:
A bounded distance decoder with decoding radius t is in most cases an incomplete decoder. Given $\underline{r} \in A^n$ it produces a codeword $\underline{v}$ if and only if $d_H(\underline{r},\underline{v}) \leq t$. Said another way:

$$For\ \underline{v} \in C, \underline{r} \in A^n,\ d_H(\underline{r},\underline{v}) \begin{cases} if \leq t \Rightarrow decode \\ if > t \Rightarrow failure \end{cases}$$



Note that an incomplete decoder is not necessarily inferior to a complete decoder. In cases where retransmission (ARQ or hybrid schemes) is conceivable and perhaps complete FEC is too costly, and incomplete decoder is the way to go, so incomplete decoders are really answers to specific trade off scenarios by combining error detection and error correction capabilities.

6 Abstract Algebra

So far our treatment of block codes has been very general. Without imposing further structure on the codes, i.e. without considering special classes of codes, there is very little further to say. One very useful class of block codes is the linear class. Linear codes have very rich algebraic structures which lend themselves to the design of encoders and decoders for them. Linear code design was pioneered by Hamming, Golay, and Berlekamp in the 40s. For the following two decades, various significant and useful discoveries ensued from this infusion of algebra in to coding theory. The 1968 textbook: "Algebraic Coding Theory" by Elwyn Berlekamp remains a keystone textbook on the subject.

Abstract algebra will allow us to define codes, define decoders, and actually decode, i.e. when $\underline{m}$ arrives, it is too much work to compare it to every $\underline{c_i} \in C$. Hence algebra lets us make fast decisions. Note that there are many perspectives in which the following algebra can be discussed. We will use the engineering perspective.

Definition: A binary operation $\cdot$ on a set X is a mapping: $X \times X \rightarrow X$ (i.e. closure is automatic)

Examples:
1. $\{0, 1, 2, ...\}$ with $\cdot$ as conventional addition.
2. $\mathbb{Z}$ with $\cdot$ as conventional multiplication.
3. The set of 3×3 matrices with $\cdot$ as conventional matrix multiplication.

Definitions:

Semigroup: a set $\hat{G}$ with an associative binary operation.

Associative: $\forall x, y, z, \in \hat{G} : (x \cdot y) \cdot z = x \cdot (y \cdot z)$.

Monoid: a semigroup enjoying a unique identity element e.
$\exists e \in \hat{G} : e \cdot x = x \cdot e = x$

Commutative Semigroup: a semigroup enjoying commutativity.

Commutative Monoid: a monoid enjoying commutativity.

Commutative: $x \cdot y = y \cdot x$.

Group: a monoid with inverse elements.
$\forall x \in \hat{G}, \exists x^{-1} : x \cdot x^{-1} = x^{-1} \cdot x = e$
$x^{-1} =$ inverse of x

Abelain or Commutative Groups: a group enjoying commutativity.

Remark: A group is a set and an operation satisfying the following four axioms: closure, associativity, identity element, inverse element.