Computer Science 1001.py

Lecture 23[†]:
Error Detection and Correction Codes (cont.)
Hamming $(7, 4, 3)$ Code

Instructors: Benny Chor, Amir Rubinstein
Teaching Assistants: Yael Baran, Michal Kleinbort

Founding Teaching Assistant (and Python Guru): Rani Hod

School of Computer Science
Tel-Aviv University, Spring Semester, 2015
http://tau-cs1001-py.wikidot.com

Lecture 22: Topics

- Simple error correction codes and error detection codes.
- Hamming distance.
- The binary symmetric channel.
- Intuition regarding the geometry and of codes.

Lecture 23: Plan

- Spheres around codewords.
- Hamming $(7, 4, 3)$ code.

The Teaching Survey

- Tel-Aviv University runs an on-line teaching survey during the last 3 weeks of each semester (*i.e.* now).
- We encourage each of you to take advantage of this opportunity.
- It is anonymous.
- Filling it even gives you a few extra points in next semester's bidding.
- However, you won't get any bonus in the grade for filling it (anonymity, right :-)
- In addition to ticking boxes, there is also space for "free style" comments.

- We do read the survey as well as individual comments, and seriously take them into account in the next round(s) of the course.

Definitions and Properties (reminder)

An encoding, E, from k to n ($k < n$) bits, is a one-to-one mapping $E : \{0,1\}^k \mapsto \{0,1\}^n$.

The set of codewords is $C = \{y \in \{0,1\}^n \mid \exists x \in \{0,1\}^k, E(x) = y\}$.
The set C is often called the code.

Let $\Delta(y,z)$ denote the Hamming distance between y, z.
Let $y \in \{0,1\}^n$. The sphere of radius r around y is the set $B(y,r) = \{z \in \{0,1\}^n \mid \Delta(y,z) \leq r\}$.

The minimum distance of a code, C, is $\Delta(C) = \min_{y \neq z \in C} \{\Delta(y,z)\}$.

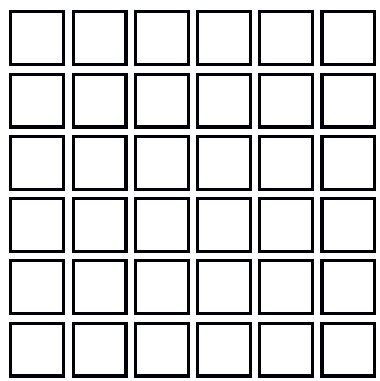
In words: The minimum distance of the code, C, is the minimal Hamming distance over all pairs of codewords in C.

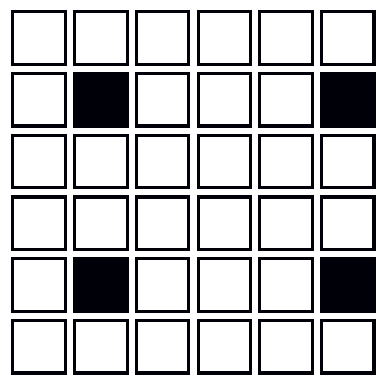
Minimum Distances of Codes (reminder)

The minimum distance of a code, C, is $\Delta(C) = \min_{y \neq z \in C} \{\Delta(y, z)\}$.

In words: The minimum distance of the code, C, is the minimal Hamming distance over all pairs of codewords in C.

- For the parity check code, $\Delta(C) = 2$.
- For the Isareli ID code, $\Delta(C) = 2$.
- For the repetition code (the case of 3 copies), $\Delta(C) = 3$.
- For the 6-by-6 cards codes, $\Delta(C) = 4$.





Closest Codeword Decoding (reminder)

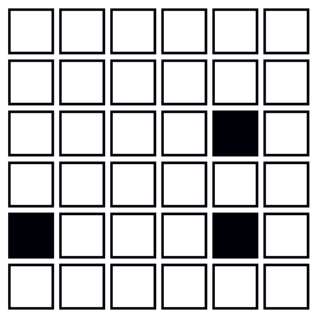
Given a code $C \subset \{0,1\}^n$ and an element $t \in \{0,1\}^n$, the closest codeword decoding, D, maps the element t to a codeword $y \in C$, that minimizes the distance $\Delta(t,z)_{z \in C}$.

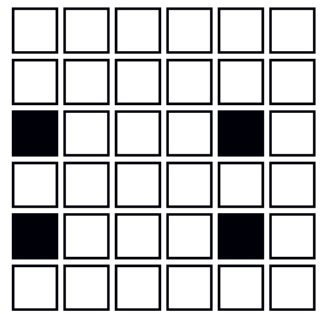
We say that such $y \in C$ is the decoding of $t \in \{0,1\}^n$.

If there is more than one $y \in C$ that attains the minimum distance, $D(t)$ announces an error.

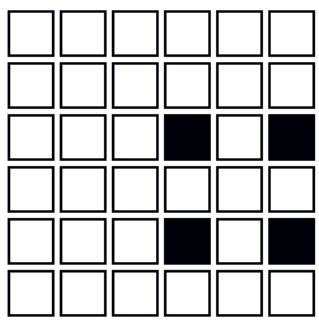
Closest Codeword Decoding: Example

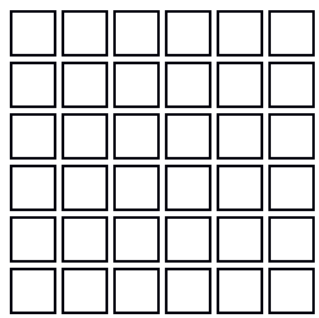
In the card magic code, suppose we receive the following string over $\{0,1\}^{36}$, depicted pictorially as the 6-by-6 black and white matrix on the left. There is a single codeword at distance 1 from this string, depicted to the right.





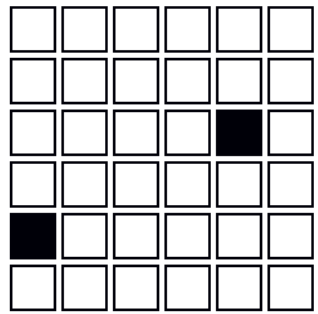
There is no codeword at distance 2 from this string (why?), and many that are at distance 3. Some of those are shown below.



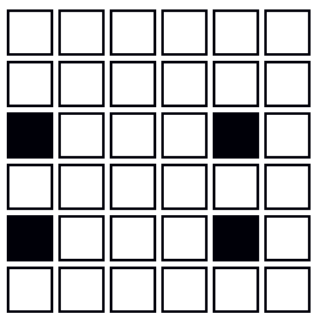


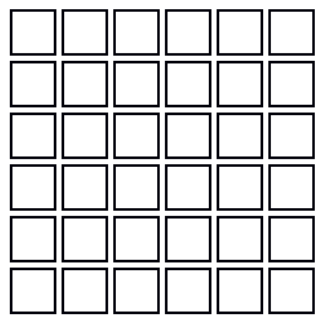
Closest Codeword Decoding: Example 2

In the card magic code, suppose we receive the following string over $\{0,1\}^{36}$, depicted pictorially as the 6-by-6 black and white matrix.



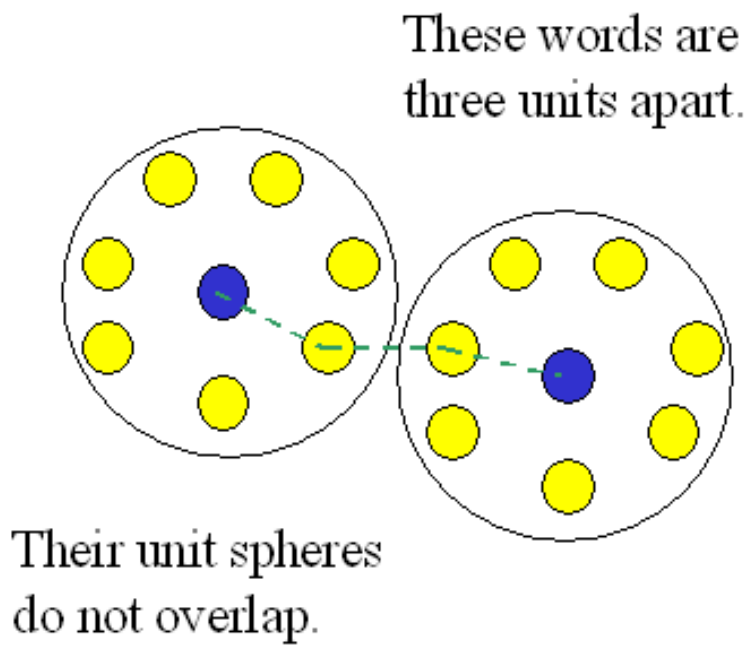
There is no codeword at distance 1 from this string (why?). There are exactly two codewords that are at distance 2 from this string. They are shown below. In such situation, closest codeword decoding announces an error.





An Important Geometric Property (reminder)

Proposition:
Suppose $d = \Delta(C)$ is the minimum distance of a code, C.
Then this code is capable of detecting up to $d - 1$ errors, and correcting up to $\lfloor (d-1)/2 \rfloor$ errors.



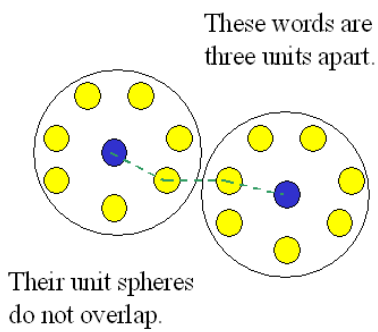
(figure from course EE 387, by John Gill, Stanford University, 2010.)

An Important Geometric Property, cont. (reminder)

Proposition:
Suppose $d = \Delta(C)$ is the minimum distance of a code, C.
Then this code is capable of detecting up to $d - 1$ errors.

Proof Idea: Let $y \in C \subset \{0,1\}^n$ be a codeword. Suppose it experienced h errors, where $1 \leq h \leq d - 1$. In other words, y was sent, and z was received, where the Hamming distance between y and z is h.

The minimum distance of the code, C, is d. Therefore z cannot be a codeword. The receiving side can detect this fact.

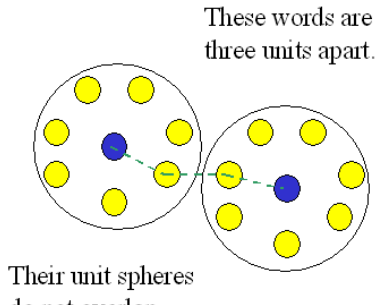


An Important Geometric Property, cont. cont. (reminder)

Proposition:
Suppose $d = \Delta(C)$ is the minimum distance of a code, C.
Then this code is capable of correcting up to $\lfloor (d-1)/2 \rfloor$ errors.

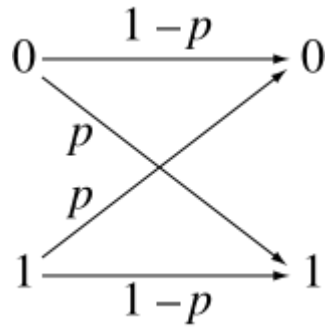
Proof Idea: Let $y \in C \subset \{0,1\}^n$ be a codeword. Suppose it experienced h errors, where $1 \leq h \leq \lfloor (d-1)/2 \rfloor$. In other words, y was sent, and z was received, where the Hamming distance between y and z is h.

The minimum distance of the code, C, is d. Therefore z cannot be within a distance of $\lfloor (d-1)/2 \rfloor$ from another codeword, t (if it were, then by the triangle inequality, the distance between the codewords y and z would be at most $\lfloor (d-1)/2 \rfloor + \lfloor (d-1)/2 \rfloor = d-1$).
Therefore, the closest codeword to z is y. The recieving side can thus decode z as y, and this decoding is correct.



Closest Codeword is Maximum Likelihood Decoding (for reference only)

Observation: For the binary symmetric channel ($p < 1/2$), closest codeword decoding of t, if defined, outputs the codeword y that maximizes the likelihood of producing t, namely $Pr(t \text{ received} \mid y \text{ sent})$.



$\forall z \neq y \in C : Pr(t \text{ received} \mid z \text{ sent}) < Pr(t \text{ received} \mid y \text{ sent})$.

Proof: Simple arithmetic, employing independence of errors hitting different bits, and $p < 1/2$.

(n, k, d) Codes: Rate and Relative Distance

Let $E : \{0,1\}^k \longrightarrow \{0,1\}^n$ be an encoding that gives rise to a code C whose minimum distance equals d.

We say that such C is an (n, k, d) code.

The rate of the code is k/n. It measures the ratio between the number of information bits, k, and transmitted bits, n.

The relative distance of the code is d/n.

- The repetition code we saw is a $(6, 2, 3)$ code. Its rate is $2/6 = 1/3$. Its relative distance is $3/6 = 1/2$.
- The parity check code is a $(3, 2, 2)$ code. Its rate and relative distance are both $2/3$.
- The card magic code is a $(36, 25, 4)$ code. Its rate is $25/36$. Its relative distance is $4/36 = 1/9$.

Goals in Code Design

- Large rate, the closer to 1 the better (smaller number of additional bits, and thus smaller communication overhead).
- Large relative distance (related to lower error in decoding).
- Efficient encoding.
- Efficient decoding (computationally).

Many useful codes, including those we saw[‡], employ linear encoding, namely the transformation $E : \{0,1\}^k \to \{0,1\}^n$ is linear $\pmod 2$. This implies that encoding can be computed as a vector-by-matrix (so called generator matrix) multiplication, making it very efficient computationally.

[‡] with the exception of the ID code.

Complexity Issues in Code Design

Many useful codes, including those we saw, employ linear encoding, namely the transformation $E : \{0,1\}^k \to \{0,1\}^n$ is linear $\mod 2$. This implies that encoding can be computed as a vector-by-matrix (so called generator matrix) multiplication, making it very efficient computationally.

Even if encoding is linear, decoding may be computationally hard for large values of k and n.

Algorithmically, decoding by the minimum distance to codeword rule may require searching over a large space of codewords, and may thus require time that is exponential in k.

This is not a problem for small values of k. For larger values, it is highly desirable to have efficient decoding as well.

Unit Balls in $\{0,1\}^n$, and their Volume

In the finite, n dimensional space over $\{0,1\}$, $\{0,1\}^n$, we define the unit ball around a point $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$,

$$B((a_1, a_2, \ldots, a_n), 1) \subseteq \{0,1\}^n \ ,$$

as the set of all points $(x_1, x_2, \ldots, x_n)$ satisfying $d\left((a_1, a_2, \ldots, a_n), (x_1, x_2, \ldots, x_n)\right) \leq 1$.

This ball is obviously a finite set. The volume of this ball is defined as the number of elements of $\{0,1\}^n$ it contains.

The number of points at distance exactly 1 from $(a_1, a_2, \ldots, a_n)$ is n. Obviously, $(a_1, a_2, \ldots, a_n)$ is also in this ball. Thus, the volume of a unit ball $B((a_1, a_2, \ldots, a_n), 1)$ in $\{0,1\}^n$ is $n+1$.

Balls of Radius r in $\{0,1\}^n$, and their Volume (reminder)

We can similarly define the ball of radius r around a point $(a_1, a_2, \ldots, a_n) \in \{0,1\}^n$,

$$B((a_1, a_2, \ldots, a_n), r) \subseteq \{0,1\}^n \ ,$$

as the set of all points $(x_1, x_2, \ldots, x_n)$ satisfying $d\left((a_1, a_2, \ldots, a_n), (x_1, x_2, \ldots, x_n)\right) \leq r$.
Without loss of generality, r is a non negative integer (otherwise just take $\lfloor r \rfloor$ to be the radius).

This ball is also a finite set. The volume of this ball is defined as the number of elements of $\{0,1\}^n$ it contains.

There are $\binom{n}{h}$ points at distance exactly h from $(a_1, a_2, \ldots, a_n)$.

Therefore, the volume of $B((a_1, a_2, \ldots, a_n), r)$, namely the number of points at distance at most r from $(a_1, a_2, \ldots, a_n)$, is $\sum_{h=0}^{r} \binom{n}{h}$.

More Definitions (for reference only)

We say that a code, C, is capable of detecting r errors if for every $y \in C$, $B(y, r) \bigcap C = \{y\}$.

We say that a code, C, is capable of correcting r errors if for every $y \neq z \in C$, $B(y, r) \bigcap B(z, r) = \emptyset$.

(do the last two definitions make sense?).

Covering $\{0,1\}^n$ by Disjoint Balls

Example: Unit balls around (0,0,0) and (1,1,1) in $\{0,1\}^n$.

Discussion using the board.

Volume Bound for $(n, k, 3)$ Codes

- Let C be a $(n, k, 3)$ code.
- This implies that balls of radius 1 around codewords are disjoint.
- Each such ball contains exactly $n + 1$ points (why?).
- There are 2^k such balls (one around each codeword).
- Since the balls are disjoint, no point in $\{0, 1\}^n$ appears twice in their union.
- Thus $2^k \cdot (n + 1) \leq 2^n$.
- The repetition code we saw is a $(6, 2, 3)$ code.
  And indeed, $2^2 \cdot (6 + 1) = 28 < 2^6 = 64$.

Volume Bound for General (n, k, d) Codes

- Let C be a (n, k, d) code. Then $2^k \cdot \sum_{\ell=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{h} \leq 2^n$.
- This is called the volume, sphere packing, or Hamming, bound.
- Example: The card magic code is a $(36, 25, 4)$ code. Indeed,

$$2^\ell \cdot \sum_{h=0}^{\lfloor (d-1)/2 \rfloor} \binom{n}{h} = 2^{25}(1 + 36) = 1,241,513,984$$

$$< 2^{36} = 68,719,476,736 \ .$$

- Proof idea: Spheres at radius $\lfloor (d-1)/2 \rfloor$ around the 2^k codewords are all disjoint. Thus the "volume" of their union cannot be greater than the "volume" of the whole space, which is 2^n.
- Codes satisfying volume bound with equality are called perfect codes.

The Singleton Bound (for reference only)

- Let C be a (n, k, d) code, then $d \leq n - k + 1$.
- This is called the Singleton bound (R.C. Singleton, 1964),
  - Proof idea: "Project" all 2^k codewords on any $k - 1$ coordinates (say the first). There are fewer combinations than number of codewords, thus at least two codewords must share the same values in all these $k - 1$ coordinates. The two codewords are at distance at least d. Thus the remaining $n - k + 1$ coordinates must have "enough room" for this distance.
- Codes that achieve equality in Singleton bound are called MDS (maximum distance separable) codes.

The Hamming $(7, 4, 3)$ Code

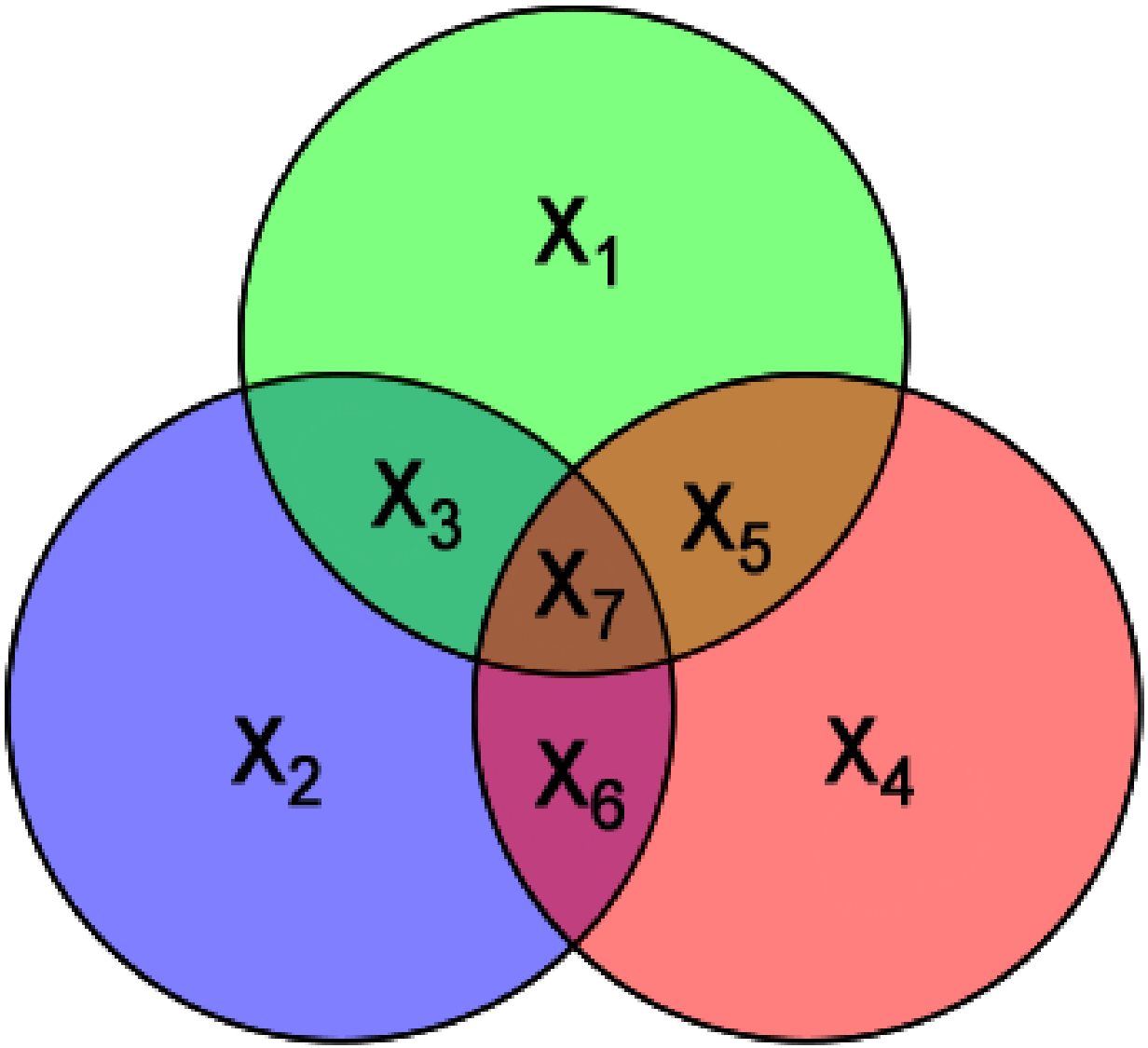
The Hamming encoder gets an original message consisting of 4 bits, and produces a 7 bit long codeword. For reasons to be clarified soon, we will number the bits in the original message in a rather unusual manner. For $(x_3, x_5, x_6, x_7) \in Z_2^4 = \{0, 1\}^4$,

$$(x_3, x_5, x_6, x_7) \longrightarrow \quad (x_1, x_2, x_3, x_4, x_5, x_6, x_7) \ ,$$

where x_1, x_2, x_4 are parity check bits, computed (modulo 2) as following:

$$\begin{aligned} x_1 &= x_3 + x_5 + x_7 \\ x_2 &= x_3 + x_6 + x_7 \\ x_4 &= x_5 + x_6 + x_7 \end{aligned}$$

Hamming $(7, 4, 3)$ Encoding: A Graphical Depiction



(modified from Wikipedia image)

Hamming Encoding in Python

This Hamming code has $2^4 = 16$ codewords. This is small enough to enable us to describe the encoding procedure using a table lookup, which in Python typically means a dictionary.

But as the size of the code grows, table lookup becomes less attractive. Besides, Hamming code, like all linear codes, has a very simple and efficient encoding procedure – each parity check bit is simply the sum modulo 2 of a subset of the information bits.

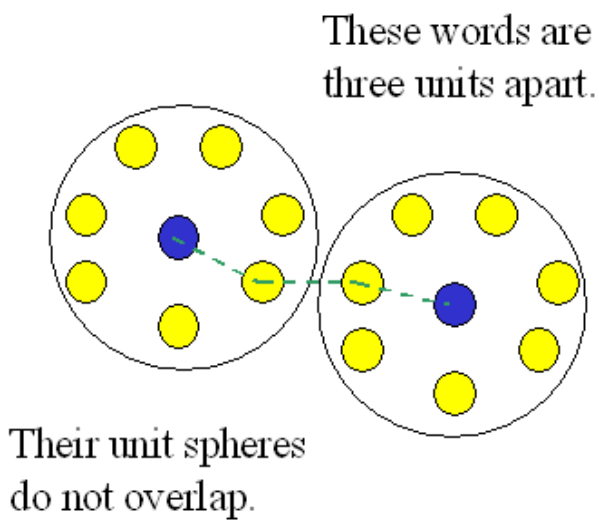
```
def hamming_encode(x3,x5,x6,x7):
    x1= (x3+x5+x7) % 2
    x2= (x3+x6+x7) % 2
    x4= (x5+x6+x7) % 2
    return (x1,x2,x3,x4,x5,x6,x7)
>>> hamming_encode(0,0,0,0)
(0, 0, 0, 0, 0, 0, 0)
>>> hamming_encode(1,0,0,0)
(1, 1, 1, 0, 0, 0, 0)
>>> hamming_encode(0,1,0,0)
(1, 0, 0, 1, 1, 0, 0)
>>> hamming_encode(0,0,1,0)
(0, 1, 0, 1, 0, 1, 0)
```

Geometry of Hamming $(7, 4, 3)$ Code

Let $\mathcal{C}_{\mathcal{H}}$ be the set of $2^4 = 16$ codewords in the Hamming code. A simple computation shows that $\mathcal{C}_{\mathcal{H}}$ equals

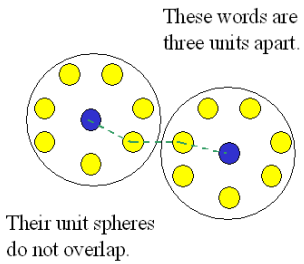
$\{(0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0),(1,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1),(0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0),(1,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1)$
$(1,\ 0,\ 0,\ 1,\ 1,\ 0,\ 0),(0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 1),(1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0),(0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1)$
$(1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0),(0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1),(1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 0),(0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1)$
$(0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0),(1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1),(0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0),(1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1)\ \}.$

By inspection, the Hamming distance between two codewords is ≥ 3. Therefore the unit spheres around different codewords do not overlap.



(figure from course EE 387, by John Gill, Stanford University, 2010.)

Closest Codeword Decoding of Hamming $(7, 4, 3)$ Code



This implies that if we transmit a codeword in $\{0, 1\}^7$ and the channel changes at most one bit in the transmission (corresponding to a single error), the received word is at distance 1 from the original codeword, and its distance from any other codeword is ≥ 2.

Thus, decoding the received message by taking the closest codeword to it, guarantees to produce the original message, provided at most one error occurred.

Questions

1. How do we find the closest codeword (for our Hamming code).
2. What happens if more than a single error occurs?

Decoding Hamming $(7, 4, 3)$ Code

$k = 4$ is small enough that we can still decode exhaustively, by a table lookup (using `dict` in Python). But there is a much cooler way to decode this specific code.

Let $(y_1, y_2, y_3, y_4, y_5, y_6, y_7)$ be the 7 bit signal received by the decoder. Assume that at most one error occurred by the channel. This means that the sent message, $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ differs from the received signal in at most one location (bit).

Let the bits b_1, b_2, b_3 be defined (modulo 2) as following

$$\begin{aligned} b_1 &= y_1 + y_3 + y_5 + y_7 \\ b_2 &= y_2 + y_3 + y_6 + y_7 \\ b_3 &= y_4 + y_5 + y_6 + y_7 \end{aligned}$$

Writing the three bits (from left to right) as $b_3 b_2 b_1$, we get the binary representation of an integer ℓ in the range $\{0, 1, 2, \ldots, 7\}$.

Decoding Hamming $(7, 4, 3)$ Code

Let $(y_1, y_2, y_3, y_4, y_5, y_6, y_7)$ be the 7 bit signal received by the decoder. Assume that at most one error occurred by the channel. This means that the sent message, $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ differs from the received signal in at most one location (bit).

Let the bits b_1, b_2, b_3 be defined (modulo 2) as following

$$\begin{aligned} b_1 &= y_1 + y_3 + y_5 + y_7 \\ b_2 &= y_2 + y_3 + y_6 + y_7 \\ b_3 &= y_4 + y_5 + y_6 + y_7 \end{aligned}$$

Writing the three bits (from left to right) as $b_3 b_2 b_1$, we get the binary representation of an integer ℓ in the range $\{0, 1, 2, \ldots, 7\}$.

Decoding Rule:

- Interpret $\ell = 0$ as "no error" and return bits 3,5,6,7 of the received signal.
- Interpret other values as "error in position ℓ ", and flip y_ℓ. Return bits 3,5,6,7 of the result.

Decoding Hamming Code: Why Does It Work?

Recall the bits b_1, b_2, b_3 be defined (modulo 2) as following

$$\begin{aligned} b_1 &= y_1 + y_3 + y_5 + y_7 \\ b_2 &= y_2 + y_3 + y_6 + y_7 \\ b_3 &= y_4 + y_5 + y_6 + y_7 \end{aligned}$$

- If there was no error (for all i, $x_i = y_i$) then from the definition of x_1, x_2, x_4, it follows that $b_3 b_2 b_1$ is zero, and we correct nothing.
- If there is an error in one of the parity check bits, say $x_2 \neq y_2$. Then only the corresponding bit is non zero ($b_2 = 1$ in this case). The position ℓ ($010 \to 2$ in this case) points to the bit to be corrected.
- If there is an error in one of the original message bits, say $x_5 \neq y_5$. Then the bits of the binary representation of this location will be non zero ($b_1 = b_3 = 1$ in this case). The position ℓ ($101 \to 5$ in this case) points to the bit to be corrected.

Decoding Hamming Code: Binary Representation

Recall the Hamming encoding

$$\begin{aligned} x_1 &= x_3 + x_5 + x_7 \\ x_2 &= x_3 + x_6 + x_7 \\ x_4 &= x_5 + x_6 + x_7 \end{aligned}$$

The locations of the parity bits x_1, x_2, x_4, are all powers of two.

- x_1 corresponds to indices 3,5,7 having a 1 in the first (rightmost) position of their binary representations 011,101,111.
- x_2 corresponds to indices 3,6,7 having a 1 in the second (middle) position of their binary representations 011,110,111.
- x_4 corresponds to indices 5,6,7 having a 1 in the last (leftmost) position of their binary representations 101,110,111.

Hamming Decoding in Python

```
def hamming_decode(y1,y2,y3,y4,y5,y6,y7):
    """ Hamming decoding of the 7 bits signal """
    b1= (y1+y3+y5+y7) % 2
    b2= (y2+y3+y6+y7) % 2
    b3= (y4+y5+y6+y7) % 2
    b=4*b3+2*b2+b1 # the integer value
    if b==0 or b==1 or b==2 or b==4:
        return (y3,y5,y6,y7)
    else:
        y=[y1,y2,y3,y4,y5,y6,y7]
        y[b-1]=(y[b-1]+1) % 2  # correct bit b
        return (y[2],y[4],y[5],y[6])

>>> y=hamming_encode(0,0,1,1)
>>> z=list(y); z  # y is a tuple (immutable)
[1, 0, 0, 0, 0, 1, 1]
>>> z[6] ^=1  # ^ is XOR (flip the 7-th bit)
>>> hamming_decode(*z) * unpacks list
(0, 0, 1, 1)

>>> y=hamming_encode(0,0,1,1)
>>> z=list(y)
>>> z[0] ^=1 ; z[6] ^=1  # flip two bits
>>> hamming_decode(*z)
(0, 0, 0, 0)  # code does not correct two errors
```

Hamming (7,4,3) Code and the Volume Bound

For codes with a minimum distance $d = 3$, the volume bound is $2^k \cdot (n+1) \leq 2^n$.

In our case, $2^4 \cdot (7+1) = 128 = 2^7$.

Hence, the Hamming (7,4,3) code are all perfect. The collection of balls of radius 1 around the 2^4 codewords fill the whole space $\{0,1\}^7$.

List Decoding (for reference only)

Closest codeword decoding either produces a single codeword, or no answer at all. Assuming the latter case is rather infrequent, would it not be better to produce a short list of potential codewords instead?

This approach to error correction codes was initially explored in Elias in the late 1950s. It was revived with a sequence of algorithmic breakthroughs in the mid 1990s and in the 2000s by Sudan (1995), Guruswami and Sudan (1998), Parvaresh and Vardy (2005), and Guruswami and Rudra (2006).

List decoding is applicable (and used) in practice. It also plays an important role in complexity theory.

Error Correction/Detection: Highlights

- Using error correction codes to fight noise in communication channels.
- The binary symmetric channel.
- Three specific (families) of codes:
  - Repetition
  - Parity bit
  - Hamming
- Hamming distance and geometry.
- Spheres around codewords.
- Closest codeword decoding.

- Coding theory is a whole discipline.
- There are additional types of errors (erasures, bursts, etc.).
- And highly sophisticated codes, employing combinatorial and algebraic (finite fields) techniques.
- Numerous uses: *E.g.* communication systems, hardware design (DRAM, flash memories, etc.), and computational complexity theory.
- We have hardly scratched the surface.