[Woo21]
What are Codes?

Anakin

Updates!

Weekly updates:

- I will be presenting more of my REU work this Friday!
- Everitt Hall Room 2233 at 4PM!
- THERE IS FREE PIZZA!!

Section 1

Motivation

The Basic Problem

Let $x[1 \ldots k]$ be some (bit)string

$$x[1 \ldots k] \xrightarrow{\textsc{encode}(x)} c[1 \ldots n] \xrightarrow{\textsc{corruption}} \tilde{c}[1 \ldots n]$$

Question: Can we design ENCODE such that we can recover x from $\tilde{c}$?

In Space No One Can Hear You ~~Scream~~ Bitflip

Here's one of the most entertaining reasons why we care about this problem

- A 2003 national election in Belgium used electronic voting
- A little known candidate got more votes than were people in the town that reported an error
- A recount was done and the candidates votes decreased by $4096 = 2^{12}$
- An investigation later determined that the a bit in the magnetic cards being used got flipped due to a cosmic ray (after ruling out other causes)

Section 2

Making This Concrete

ABCs of Codes

Definition (Alphabets, Block Lengths, Codes)

A *code* C of *block length* n over a (finite) alphabet Σ is a set $C \subseteq \Sigma^n$.
An element $c \in C$ is a *code word* (big surprise here).

An Example

Consider the following encoding over $\Sigma = \{0, 1\}$ (bitstrings)

$$\begin{aligned} \textsc{Encode}\colon \{0,1\}^3 &\to \{0,1\}^4 \\ (x_1, x_2, x_3) &\mapsto (x_1, x_2, x_3, x_1 + x_2 + x_3 \pmod 2) \end{aligned}$$

$$C := \operatorname{im}(\textsc{Encode}) = \begin{Bmatrix} 0000, & 0011, & 0101, & 0110 \\ 1001, & 1010, & 1100, & 1111 \end{Bmatrix}$$

Claim: Code C can <u>correct</u> one erasure. If we lose one bit and know where it was, we can recover it.

If $\tilde{c} = 0?01$, what is c? $c = 0101$

If $\tilde{c} = 11?1$, what is c? $c = 1111$

If $\tilde{c} = 0??1$, what is c? $c = 0101$ or $c = 0011$

An Example

Consider the following encoding over $\Sigma = \{0, 1\}$ (bitstrings)

$$\begin{aligned} \textsc{encode}\colon \{0,1\}^3 &\to \{0,1\}^4 \\ (x_1, x_2, x_3) &\mapsto (x_1, x_2, x_3, x_1 + x_2 + x_3 \pmod 2) \end{aligned}$$

$$C := \operatorname{im}(\textsc{encode}) = \left\{ \begin{matrix} (0,0,0,0), & (0,0,1,1), & (0,1,0,1), & (0,1,1,0), \\ (1,0,0,1), & (1,0,1,0), & (1,1,0,0), & (1,1,1,1) \end{matrix} \right\}$$

Claim: Code C can <u>detect</u> one error. We can tell detect with certainty if $\tilde{c} \in C$ or $\tilde{c} \notin C$ if at most one bit was flipped

If $\tilde{c} = 0001$, was there an error? YES! what is c?
$c \in \{0000, 1001, 0101, 0011\}$

If $\tilde{c} = 0000$, was there an error? Who knows? c could be 0000 or 0110

Metrics and Geometry

Consider the following encoding over $\Sigma = \{0,1\}$ and $C \coloneqq \operatorname{im}(\textsc{encode})$

$\textsc{encode}\colon \{0,1\}^4 \to \{0,1\}^7$

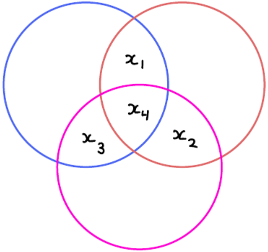
$$(x_1,x_2,x_3,x_4) \mapsto (x_1,x_2,x_3,x_4,\overbrace{x_2+x_3+x_4, x_1+x_3+x_4, x_1+x_2+x_4}^{\text{mod } 2})$$

Claim: This code can correct one error! So we can tell if $\tilde{c} \in C$ or if $\tilde{c} \notin C$ and there is one flipped bit, we can correct it.

We will show this geometrically

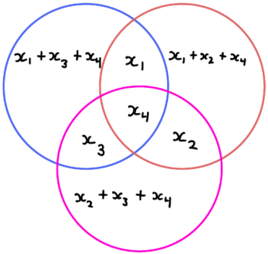
$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, \overbrace{x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4}^{\text{mod } 2})$$

Question: If $\tilde{c} = 0111010$, what is c?



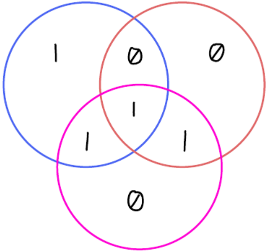
$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, \overbrace{x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4}^{\text{mod } 2})$$

Question: If $\tilde{c} = 0111010$, what is c?



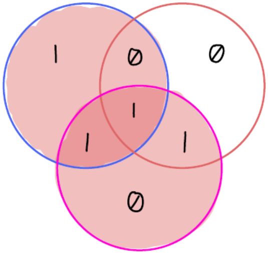
$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, \overbrace{x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4}^{\text{mod } 2})$$

Question: If $\tilde{c} = 0111010$, what is c?



$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, \overbrace{x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4}^{\text{mod } 2})$$

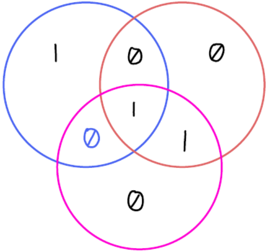
Question: If $\tilde{c} = 0111010$, what is c?



$$(x_1, x_2, x_3, x_4) \mapsto (x_1, x_2, x_3, x_4, \overbrace{x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4}^{\text{mod } 2})$$

Question: If $\tilde{c} = 0111010$, what is c?

Answer: $c = 0101010$



Formalizing What We Just Saw

The code we just saw is an example of the *Hamming Code*.

Definition ((Relative) Hamming Distance)

The *Hamming Distance* $\Delta(x, y)$ between $x, y \in \Sigma^n$ is the number of positions where x and y have different characters.

Exercise: If you know what a metric is, this is a metric. $\forall x, y, z \in \Sigma^n$:

- $\Delta(x, x) = 0$, $\Delta(x, y) = \Delta(y, x)$, If $x \neq y$, $\Delta(x, y) > 0$
- $\Delta(x, z) \leq \Delta(x, y) + \Delta(y, z)$ (Triangle Inequality)

The *Relative Hamming Distance* $\delta(x, y)$ is $\frac{\Delta(x,y)}{n}$.

Definition (Minimum Distance)

The *Minimum Distance* of a code C is $\min\limits_{x \neq y \in C} \Delta(x, y)$

Robustness of a Code

Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

Examples: (Go check these!)

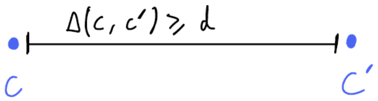
- The first code we saw $\textsc{Encode}: \Sigma^3 \to \Sigma^4$ has distance 2
- The second code we saw $\textsc{Encode}: \Sigma^4 \to \Sigma^7$ has distance 3

Proof: Pretty Pictures

Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors



Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

c $\qquad\qquad$ c'

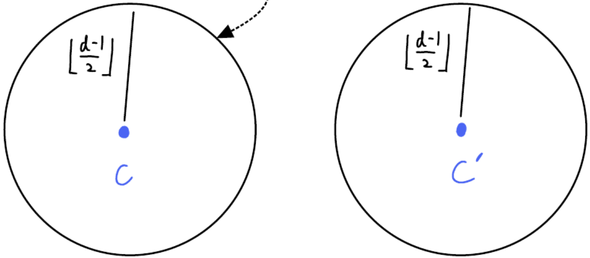
$\textsc{Correct}(\tilde{c})$:
 return $c \in C$ minimizing $\Delta(c, \tilde{c})$

Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

$$\left\{ x \in \Sigma^n \;\middle|\; \Delta(x, c) \leq \left\lfloor \frac{d-1}{2} \right\rfloor \right\}$$

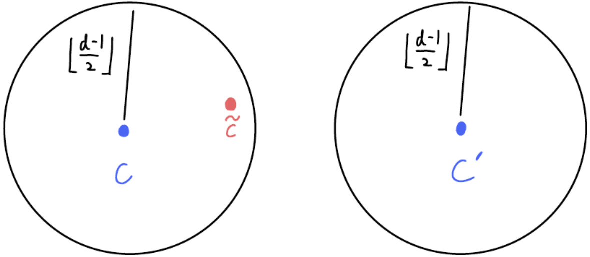


$\textsc{Correct}(\tilde{c})$:
  return $c \in C$ minimizing $\Delta(c, \tilde{c})$

Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors



$\textsc{Correct}(\tilde{c})$:
return $c \in C$ minimizing $\Delta(c, \tilde{c})$

Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\lfloor \frac{d-1}{2} \rfloor$ errors

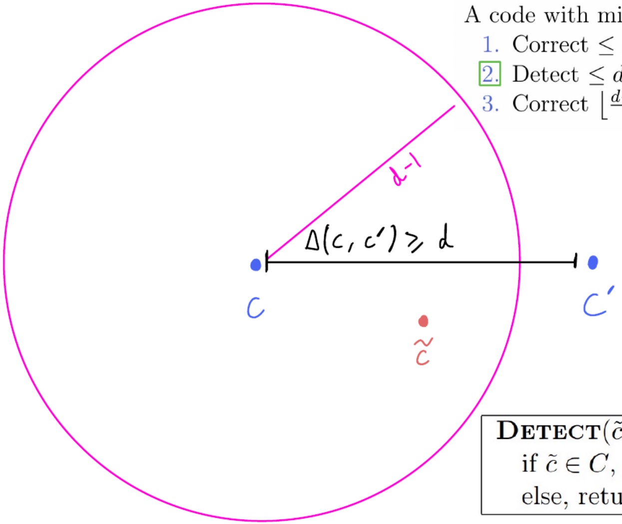
c

c'

$\textsc{Detect}(\tilde{c})$:

if $\tilde{c} \in C$, return "no error"

else, return "error"



Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

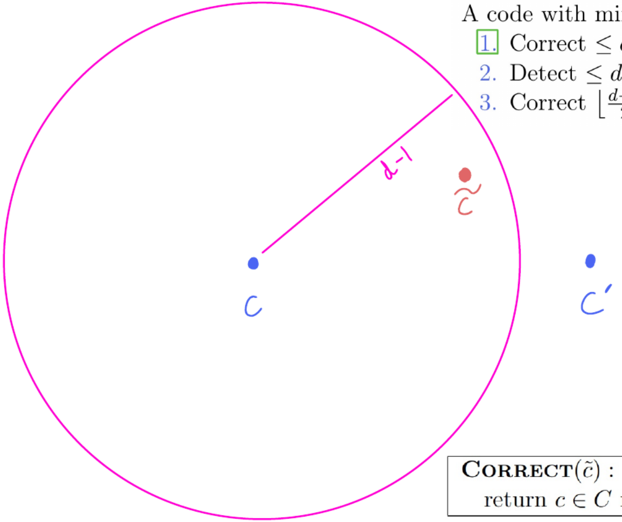
Detect$(\tilde{c})$:
  if $\tilde{c} \in C$, return "no error"
  else, return "error"

What is an erasure?

$$
\begin{array}{lcccccccc}
c = & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \cdots \\
\tilde{c} = & x_1 & x_2 & \_ & \_ & x_5 & \_ & x_7 & x_8 \cdots
\end{array}
$$

Question: If there are $\leq d-1$ erasures, what is the max value of $\Delta(c, \tilde{c})$?

Answer: $d-1$!



Theorem

A code with minimum distance d can

1. Correct $\leq d-1$ erasures
2. Detect $\leq d-1$ errors
3. Correct $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

$\textsc{Correct}(\tilde{c})$:
  return $c \in C$ minimizing $\Delta(c, \tilde{c})$

Sadly, Pictures are Misleading

You may ask "If $\tilde{c}$ is of distance $\leq d-1$ to c and c' can be as close as c' then could $\tilde{c}$ be closer to c'?"

- Suppose that $\tilde{c}$ is a corruption of c and has $\leq d-1$ erasures. Suppose c' is a different code word to c but for contradiction, $\Delta(\tilde{c}, c') < \Delta(\tilde{c}, c)$ so $\textsc{Correct}(\tilde{c})$ incorrectly corrects $\tilde{c}$ to c'
- That means there are two different ways to fill the $\leq d-1$ erasures where one filling gives c and the other gives c'
- Since we are only dealing with erasures in $\leq d-1$, we know what the other characters are, and the other $n-d+1$ positions of c' and c must match.
- This implies $\Delta(c, c') \leq d-1 < d$ which is a contradiction to our minimum distance d, since c and c' are distinct strings. So $\textsc{Correct}(\tilde{c})$ should correctly return c

Efficiency and Overhead

Definition (Message Length)

The *message length / dimension* of a code C over an alphabet Σ is $k := \log_{|\Sigma|} |C|$

Remember we were talking at the beginning of encoding messages of length k into a code C of messages of length n? This is the same k?

- Our messages live in Σ^k and get mapped to a code in C
- We want every code word in C to correspond to exactly one message and every message to map to exactly one code word
  - Want $|C| = \left|\Sigma^k\right| = \left|\Sigma^k\right|$
- Rearranging yields $k = \log_{|\Sigma|} |C|$

Efficiency and Overhead

Definition (Rate)

The *rate* of a code $C \subseteq \Sigma^n$ is $R = \frac{\text{message length } k}{\text{block length } n} = \frac{\log_{|\Sigma|} |C|}{n}$

- $R \in [0, 1]$
- This is sort of the measure of the efficiency of the code
- R close to 1 means message does not grow that much after encoding
- R close to 0 means messages grows quite alot

Trade-offs

Consider an encoding $x \mapsto c$ which perhaps gets corrupted into $\tilde{c}$

- We want to handle when something bad happens to c
- We want to recover information about x from $c/\tilde{c}$

Trade-offs

Consider an encoding $x \mapsto c$ which perhaps gets corrupted into $\tilde{c}$

- We want distance d
- We want to minimize overhead

Trade-offs

Consider an encoding $x \mapsto c$ which perhaps gets corrupted into $\tilde{c}$

- We want distance d
- We want rate as close to 1 as possible

Motivating Question: What is the trade-off between distance and rate?

Questions?

Information is the resolution of uncertainty.

— CLAUDE E SHANNON (1948)

Bibliography

Mary Wootters.

Lecture 1 video 2: Definitions and examples, 2021.