Error Correcting Codes
Welcome to the first class of this course.
You are in for a fun ride! This semester!

What do these sentences say?

Why did this work?

Redundancy!

Codes are clever ways of judiciously adding redundancy to enable recovery under “noise”.
General Model

“Noise” introduced by the channel:
- changed fields in the codeword vector (e.g. a flipped bit).
  - Called **errors**
- missing fields in the codeword vector (e.g. a lost byte).
  - Called **erasures**

How the decoder deals with errors and/or erasures?
- **detection** (only needed for errors)
- **correction**
Applications

• **Storage**: CDs, DVDs, hard disks, Flash,…
• **Wireless**: Cell phones, wireless links,..
• **Satellite and Space**: TV, Mars rover, …
• **Digital Television**: DVD, MPEG2 layover,

**Reed-Solomon** codes were traditionally the most used in practice.

**LDPC** codes used for 4G (and 5G) communication. Algorithms for decoding are quite sophisticated.
Block Codes

symbols (e.g., bits)

block 1

message 1

block 2

message 2

Other kind: convolutional codes (we won’t cover it)…
Block Codes

- Each message and codeword is of fixed size
- Notation:

  \[ k = |m| \]
  length of the message

  \[ n = |c| \]
  length of the codeword

  \( C = \text{"code"} = \text{set of codewords} \)
Simple Examples

3-Replication code: k=1, n=3

<board>

• How many **erasures** can be recovered?
• How many **errors** can be detected?
• Up to how many **errors** can be corrected?

**Errors are much harder to deal with than erasures.**

Why?

Need to find out **where** the errors are!
Simple Examples

Single parity check code: $k=2$, $n=3$

Consider codewords as vertices on a hypercube.

- codeword

$n = 3 = \text{dimensionality}$

$2^n = 8 = \text{number of nodes}$
Simple Examples

Single parity check code: $k=2$, $n=3$

- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Cannot even correct single error. Why?

**Codewords are too “close by”**

Let’s formalize this notion of distance..
In general, symbols come from an “alphabet”.

**Notation:**
\[ \Sigma = \text{alphabet} \]
\[ q = |\Sigma| = \text{alphabet size} \]

**Question:**
What alphabet did we use so far?

**C ⊆ \Sigma^n (codewords)**
Block Codes

Notion of distance between codewords:

$$
\Delta(x,y) = \text{number of positions s.t. } x_i \neq y_i
$$

minimum distance of a code

$$
d = \min\{\Delta(x,y) : x,y \in C, x \neq y\}
$$

Code described as: $$(n, k, d)_q$$
Binary Codes

Today we will mostly be considering $\sum = \{0, 1\}$ and will sometimes use $(n, k, d)$ as shorthand for $(n, k, d)_2$

In binary $\Delta(x, y) = |\{ i : x_i \neq y_i \}|$

is often called the **Hamming distance**
Example of (6,3,3)\textsubscript{2} systematic code

<table>
<thead>
<tr>
<th>message</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000000</td>
</tr>
<tr>
<td>001</td>
<td>001011</td>
</tr>
<tr>
<td>010</td>
<td>010101</td>
</tr>
<tr>
<td>011</td>
<td>011110</td>
</tr>
<tr>
<td>100</td>
<td>100110</td>
</tr>
<tr>
<td>101</td>
<td>101101</td>
</tr>
<tr>
<td>110</td>
<td>110011</td>
</tr>
<tr>
<td>111</td>
<td>111000</td>
</tr>
</tbody>
</table>

**Definition:** A Systematic code is one in which the message appears in the codeword.
Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?

2 bits
0 -> 00, 1 -> 11
(n=2, k=1, d=2)

3 bits
0 -> 000, 1 -> 111
(n=3, k=1, d=3)

In general need \( d \geq 3 \) to correct one error. Why?
Role of Minimum Distance

Theorem:
A code C with minimum distance “d” can:
   1. detect any \((d-1)\) errors
   2. recover any \((d-1)\) erasures
   3. correct any <write> errors

Proof: <Will be part of homework>
Desiderata

We look for codes with the following properties:

1. Good rate: $k/n$ should be high (low overhead)
2. Good distance: $d$ should be large (good error correction)
3. Small block size $k$
4. Fast encoding and decoding
5. Others: want to handle bursty/random errors, local decodability, ...
We will begin next class with Hamming Codes