15-853:Algorithms in the Real World

Error Correcting Codes

Welc**e t* t*e fi*st clas* o* t*is course.
Y*u a** in f*r a f*n rid* th*s se*est*r!

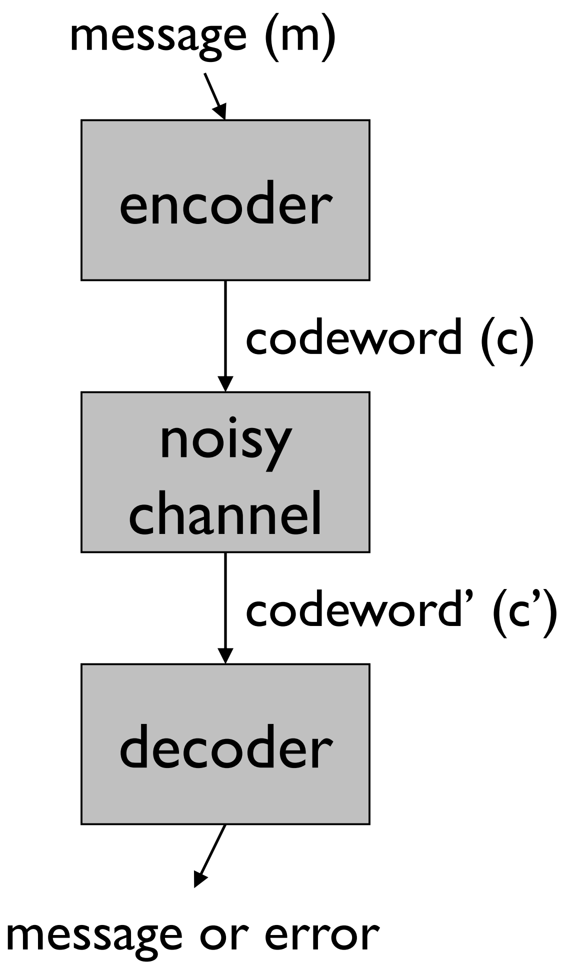
What do these sentences say?

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

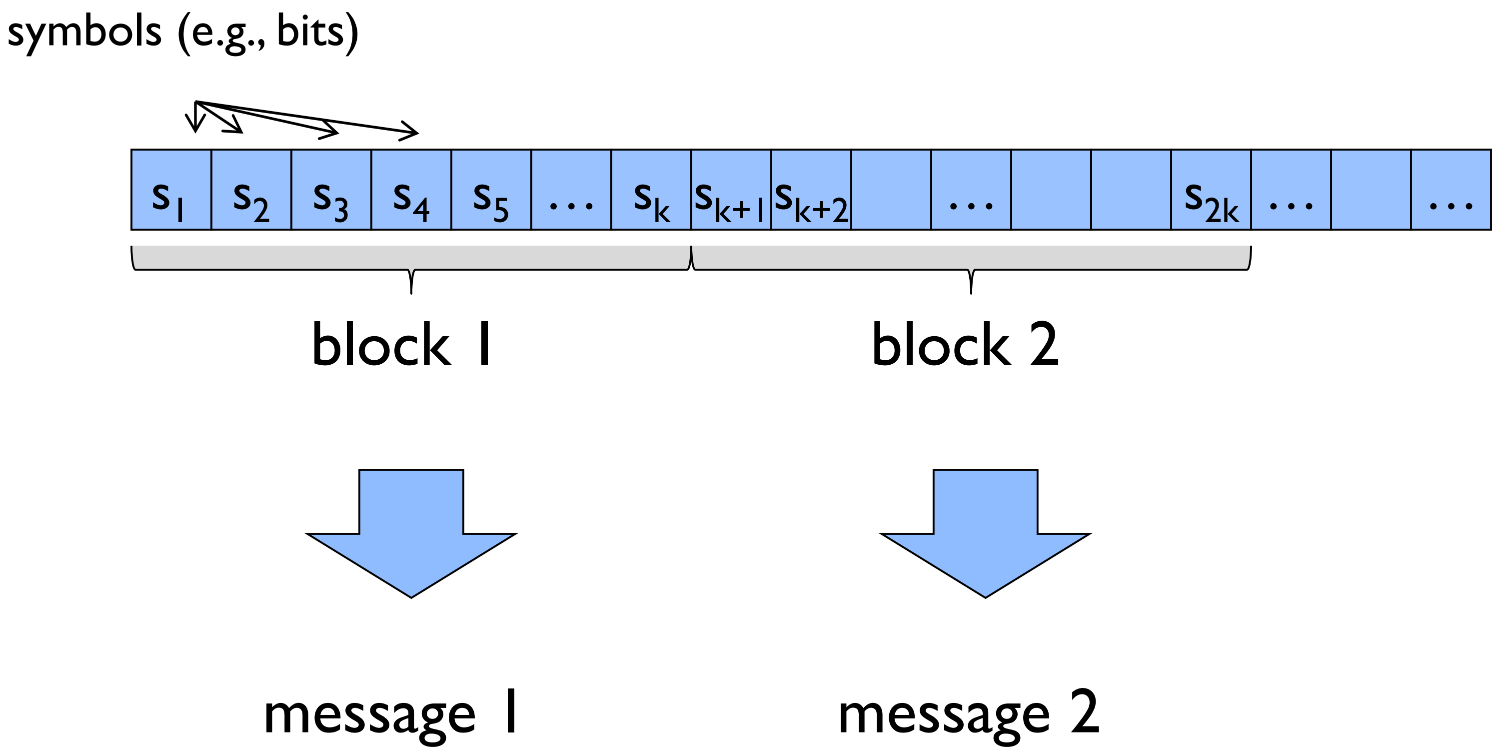
- **Storage**: CDs, DVDs, hard disks, Flash,…
- **Wireless**: Cell phones, wireless links,..
- **Satellite and Space**: TV, Mars rover, …
- **Digital Television**: DVD, MPEG2 layover,

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

LDPC codes used for 4G (and 5G) communication.

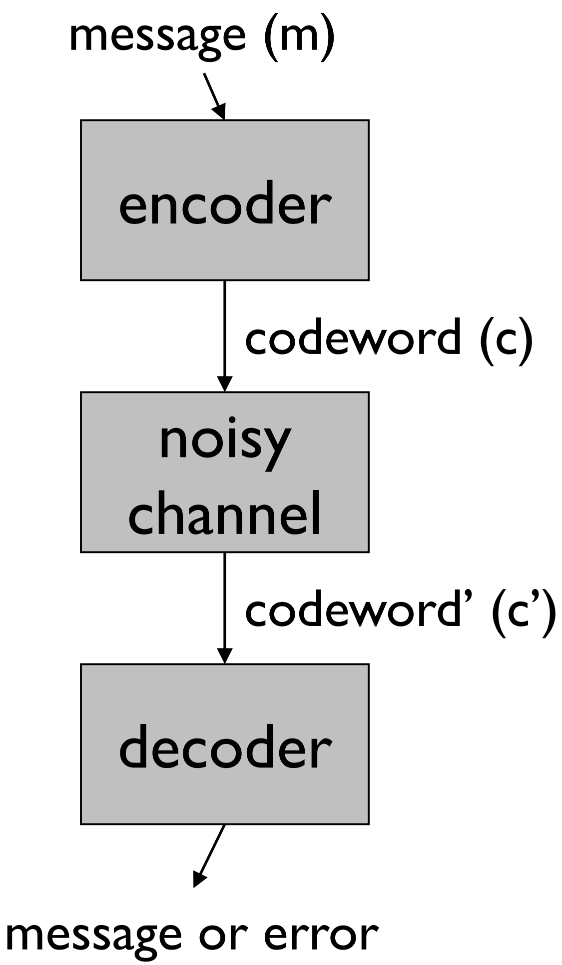
Algorithms for decoding are quite sophisticated.

Block Codes



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 = "code" = set of codewords

Simple Examples

3-Repetition 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?

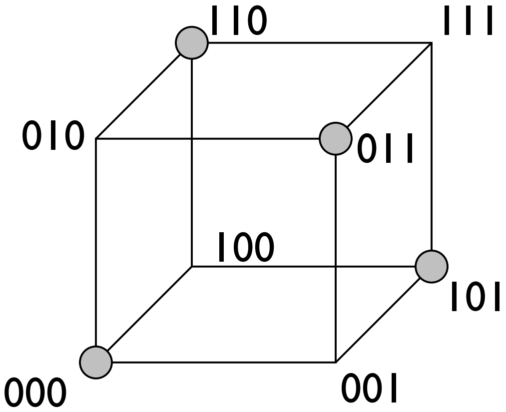
Need to find out **where** the errors are!

Simple Examples

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

<board>

Consider codewords as vertices on a hypercube.



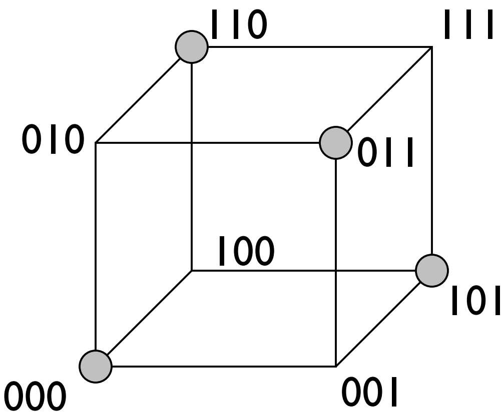
● codeword

n = 3 = dimensionality

2^n = 8 = number of nodes

Simple Examples

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



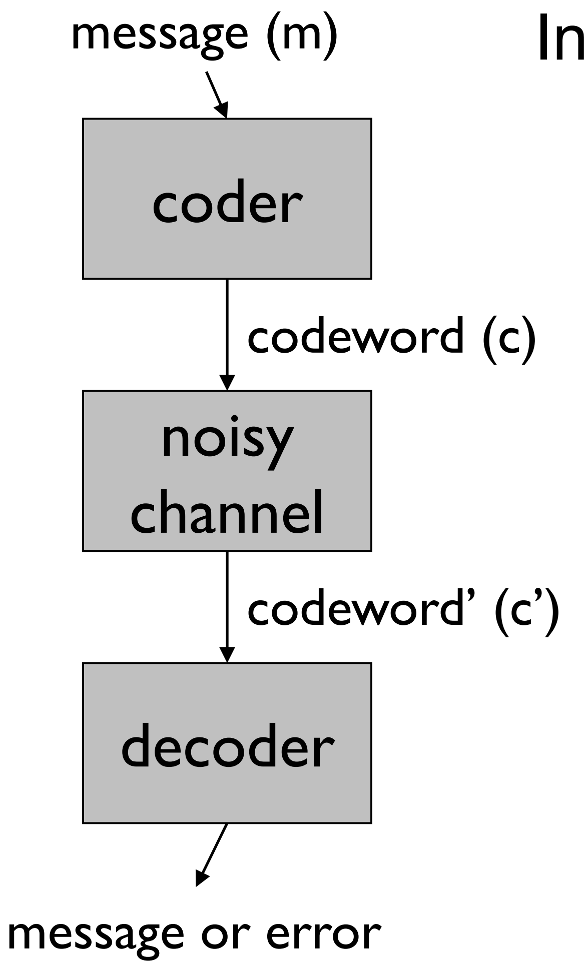
- How many **erasures** can be recovered?
- How many **errors** can be **detected**?
- Up to how many **errors** can be **corrected**?

Cannot even correct single error. Why?

Codewords are too “close by”

Let’s formalize this notion of distance..

Block Codes



In general, symbols come from an "**alphabet**"

Notation:

Σ = alphabet

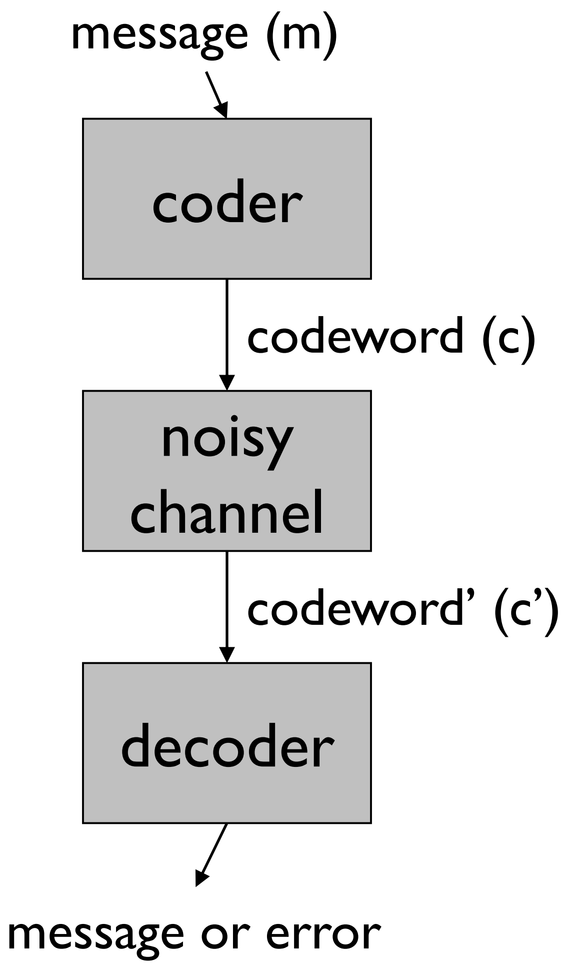
q = $|\Sigma|$ = alphabet size

Question:

What alphabet did we use so far?

C $\subseteq \Sigma^n$ (codewords)

Block Codes



Notion of distance between codewords:

Δ(x,y) = 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 Σ **= {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**

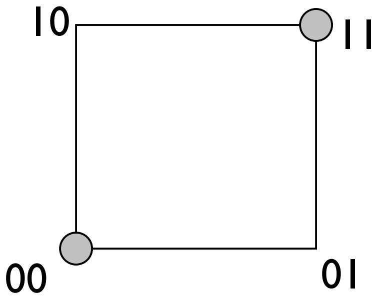
Example of $(6,3,3)_2$ systematic code

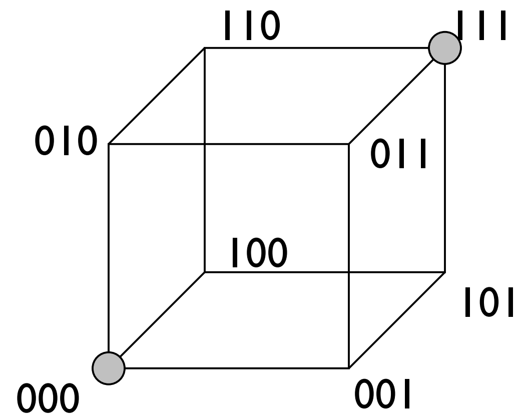
message	codeword
000	**000**000
001	**001**011
010	**010**101
011	**011**110
100	**100**110
101	**101**101
110	**110**011
111	**111**000

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>

Intuition here.. minimum-distance-decoding..

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