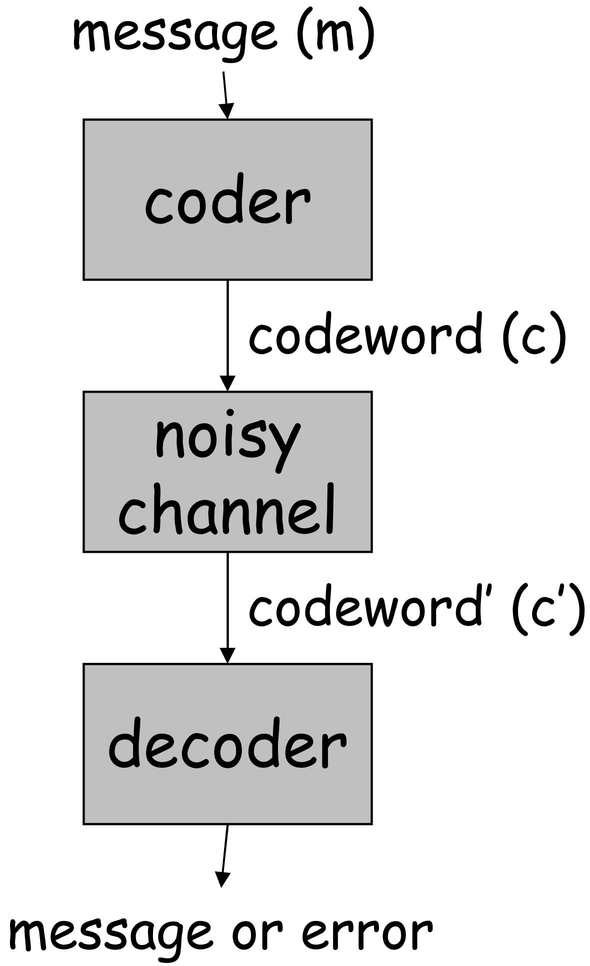
Error Correcting Codes

- Overview

– Hamming Codes

– Linear Codes

General Model



Errors introduced by the noisy channel:

- changed fields in the codeword (e.g. a flipped bit)
- missing fields in the codeword (e.g. a lost byte). Called **erasures**

How the decoder deals with errors.

- **error detection** vs.
- **error correction**

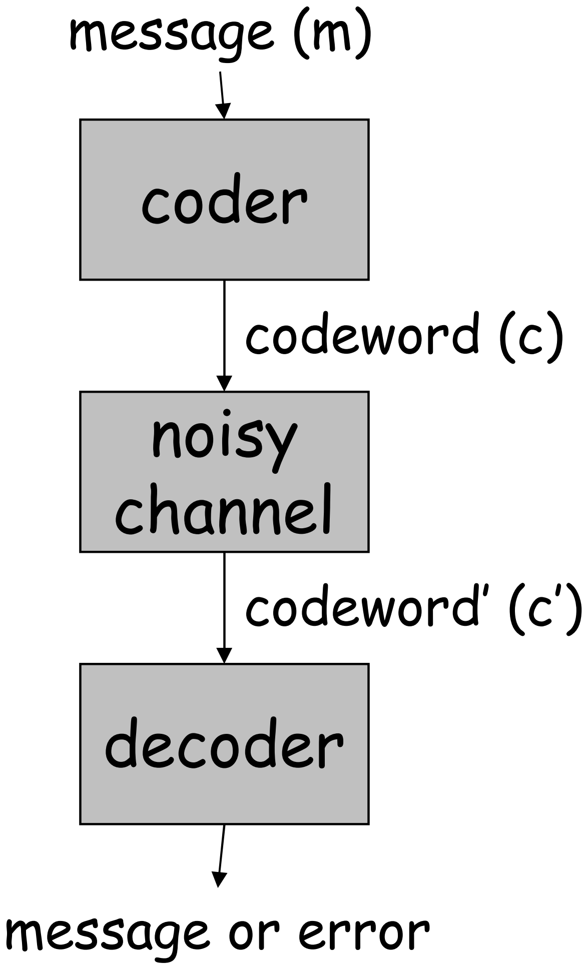
Applications

- **Storage**: CDs, DVDs, "hard drives",
- **Wireless**: Cell phones, wireless links
- **Satellite and Space**: TV, Mars rover, …
- **Digital Television**: DVD, MPEG2 layover
- **High Speed Modems**: ADSL, DSL, ..

Reed-Solomon codes are by far the most used in practice, including pretty much all the examples mentioned above.

Algorithms for decoding are quite sophisticated.

Block Codes



Each message and codeword is of fixed size

Σ = codeword alphabet

k = |m| **n** = |c| **q** = |Σ|

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

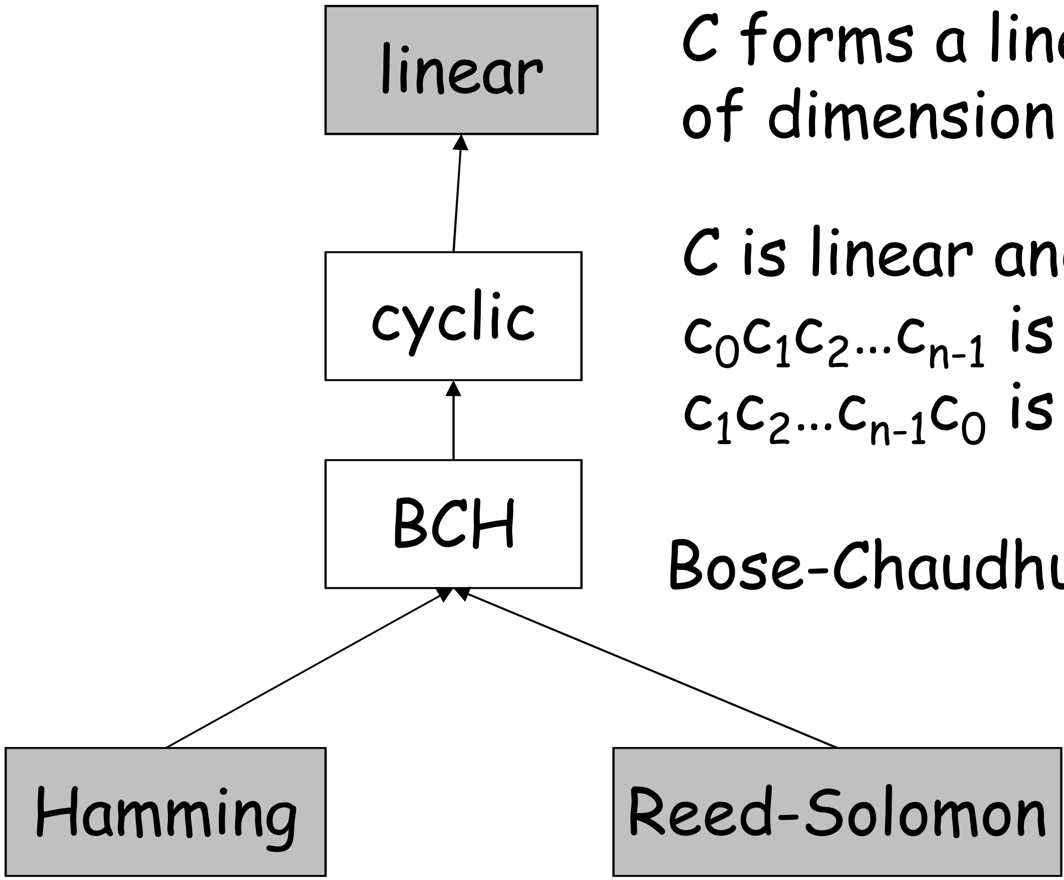
Δ**(x,y)** = number of positions s.t. $x_i \neq y_i$

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

s = max{Δ(c,c')} that the code can correct

Code described as: $(\mathbf{n,k,d})_q$

Hierarchy of Codes



C forms a linear subspace of Σ^n of dimension k

C is linear and
$c_0c_1c_2\ldots c_{n-1}$ is a codeword implies
$c_1c_2\ldots c_{n-1}c_0$ is a codeword

Bose-Chaudhuri-Hochquenghem

These are all block codes (operate on fixed-length strengths).

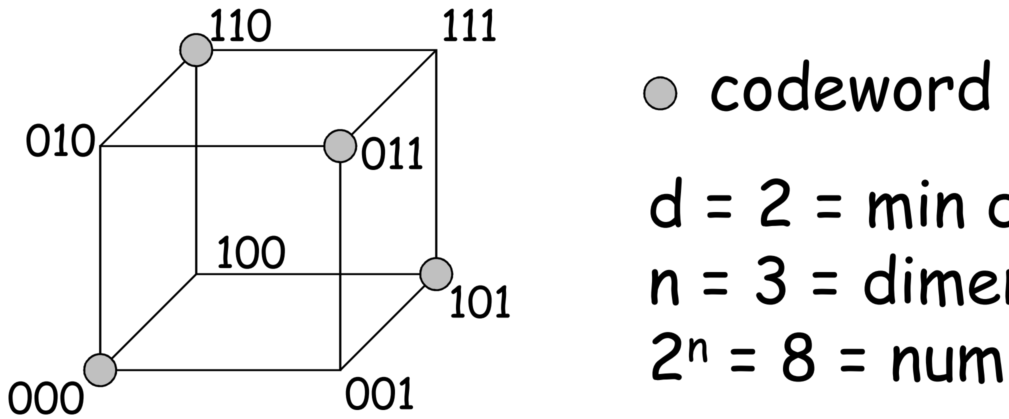
Binary Codes

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 $\Delta(x,y)$ is often called the **Hamming distance**

Hypercube Interpretation

Consider codewords as vertices on a hypercube.



d = 2 = min distance
n = 3 = dimensionality
2^n = 8 = number of nodes

The distance between nodes on the hypercube is the Hamming distance Δ. The minimum distance is d.

001 is equidistance from 000, 011 and 101.

For s-bit error detection d ≥ s + 1

For s-bit error correction d ≥ 2s + 1

Error Detection with Parity Bit

A $(k+1,k,2)_2$ code

Encoding:

$$m_1 m_2 \ldots m_k \Rightarrow m_1 m_2 \ldots m_k p_{k+1}$$

where $p_{k+1} = m_1 \oplus m_2 \oplus \ldots \oplus m_k$

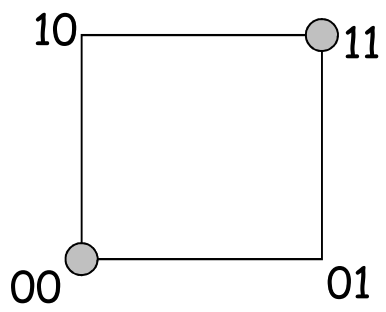
d = 2 since the parity is always even (it takes two bit changes to go from one codeword to another).

Detects one-bit error since this gives odd parity

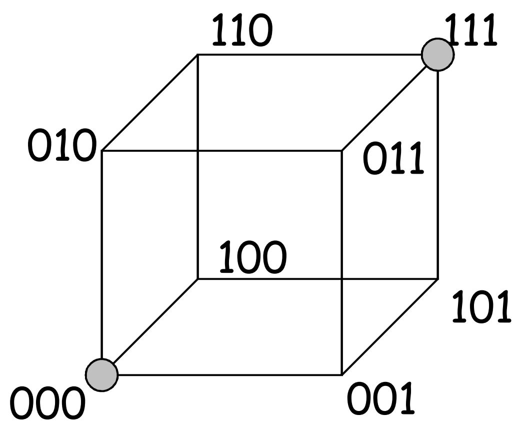
Cannot be used to correct 1-bit error since any odd-parity word is equal distance Δ to k+1 valid codewords.

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?

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

message	codeword
000	000000
001	001011
010	010101
011	011110
100	100110
101	101101
110	110011
111	111000

Definition: A **Systematic code** is one in which the message appears in the codeword

Same in any bit of message implies two bits of difference in extra codeword columns.

Error Correcting Multibit Messages

We will first discuss **Hamming Codes**

Detect and correct 1-bit errors.

Codes are of form: $(2^r-1, 2^r-1 - r, 3)$ for any $r > 1$

e.g. (3,1,3), (7,4,3), (15,11,3), (31, 26, 3), …

which correspond to 2, 3, 4, 5, … "parity bits" (i.e. n-k)

The high-level idea is to "localize" the error.

Any specific ideas?

Hamming Codes: Encoding

Localizing error to top or bottom half 1xxx or 0xxx

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5		m_3			p_0

$$p_8 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$$

Localizing error to x1xx or x0xx

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3			p_0

$$p_4 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$$

Localizing error to xx1x or xx0x

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2		p_0

$$p_2 = m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$$

Localizing error to xxx1 or xxx0

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2	p_1	p_0

$$p_1 = m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$$

Hamming Codes: Decoding

m_{15}	m_{14}	m_{13}	m_{12}	m_{11}	m_{10}	m_9	p_8	m_7	m_6	m_5	p_4	m_3	p_2	p_1	p_0

We don't need p_0, so we have a (15,11,?) code.

After transmission, we generate

$$b_8 = p_8 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_{11} \oplus m_{10} \oplus m_9$$
$$b_4 = p_4 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_6 \oplus m_5$$
$$b_2 = p_2 \oplus m_{15} \oplus m_{14} \oplus m_{11} \oplus m_{10} \oplus m_7 \oplus m_6 \oplus m_3$$
$$b_1 = p_1 \oplus m_{15} \oplus m_{13} \oplus m_{11} \oplus m_9 \oplus m_7 \oplus m_5 \oplus m_3$$

With no errors, these will all be zero

With one error $b_8b_4b_2b_1$ gives us the error location.

e.g. **0100** would tell us that $\mathbf{p_4}$ is wrong, and
1100 would tell us that $\mathbf{m_{12}}$ is wrong

Hamming Codes

Can be generalized to any power of 2

- $n = 2^r - 1$ (15 in the example)
- $(n-k) = r$ (4 in the example)
- $d = 3$ (discuss later)
- Can correct one error, but can't tell difference between one and two!
- Gives $(2^r-1, 2^r-1-r, 3)$ code

Extended Hamming code

- Add back the parity bit at the end
- Gives $(2^r, 2^r-1-r, 4)$ code
- Can correct one error and detect 2
- (not so obvious)

Lower bound on parity bits

How many nodes in hypercube do we need so that d = 3?

Each of the 2^k codewords eliminates n neighbors plus itself, i.e. n+1

need
$$2^n \geq (n+1)2^k$$
$$n \geq k + \log_2(n+1)$$
$$n \geq k + \lceil \log_2(n+1) \rceil$$

In previous hamming code $15 \geq 11 + \lceil \log_2(15+1) \rceil = 15$

Hamming Codes are called **perfect codes** since they match the lower bound exactly

Lower bound on parity bits

What about fixing 2 errors (i.e. d=5)?

Each of the 2^k codewords eliminates itself, its neighbors and its neighbors' neighbors, giving: $1+\binom{n}{1}+\binom{n}{2}$

$$
\begin{aligned}
2^n &\geq (1+n+n(n-1)/2)2^k \\
n &\geq k+\log_2(1+n+n(n-1)/2) \\
&\geq k+2\log_2 n-1
\end{aligned}
$$

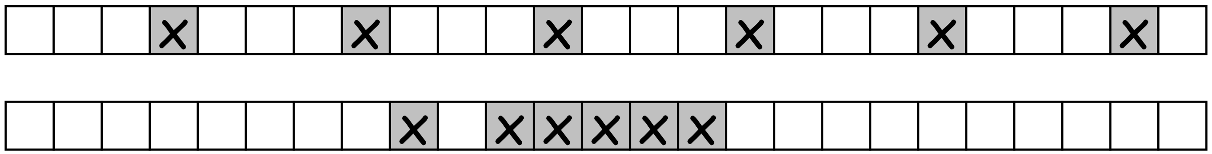
Generally to correct s errors:

$$n \geq k+\log_2(1+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{s})$$

Lower Bounds: a side note

The lower bounds assume random placement of bit errors.

In practice errors are likely to be less than random, e.g. evenly spaced or clustered:



Can we do better if we assume **regular errors**?

We will come back to this later when we talk about **Reed-Solomon** codes. In fact, this is the main reason why Reed-Solomon codes are used much more than Hamming-codes.

Linear Codes

If Σ is a field, then Σ^n is a vector space

Definition: *C* is a linear code if it is a linear subspace of Σ^n of dimension k.

This means that there is a set of k independent vectors $v_i \in \Sigma^n$ $(1 \leq i \leq k)$ that span the subspace.

i.e., every codeword can be written as:

$$c = a_1 v_1 + \ldots + a_k v_k \qquad a_i \in \Sigma$$

The sum of two codewords is a codeword.

Linear Codes

Vectors for the $(7,4,3)_2$ Hamming code:

		m_7	m_6	m_5	p_4	m_3	p_2	p_1
v_1	=	1	0	0	1	0	1	1
v_2	=	0	1	0	1	0	1	0
v_3	=	0	0	1	1	0	0	1
v_4	=	0	0	0	0	1	1	1

How can we see that $d = 3$?

Generator and Parity Check Matrices

Generator Matrix:

A k x n matrix **G** such that: $C = \{xG \mid x \in \Sigma^k\}$

Made from stacking the spanning vectors

Parity Check Matrix:

An (n – k) x n matrix **H** such that: $C = \{y \in \Sigma^n \mid Hy^T = 0\}$

Codewords are the nullspace of H

These **always exist for linear codes**

Advantages of Linear Codes

- Encoding is efficient (vector-matrix multiply)
- Error detection is efficient (vector-matrix multiply)
- **Syndrome** (Hy^T) has error information
- Gives q^{n-k} sized table for decoding
  Useful if n-k is small

Example and "Standard Form"

For the Hamming (7,4,3) code:

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

By swapping columns 4 and 5 it is in the form I_k, A.
A code with a matrix in this form is **systematic**, and
G is in "**standard form**"

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Relationship of G and H

If G is in standard form $[I_k, A]$
then $H = [A^T, I_{n-k}]$

Example of (7,4,3) Hamming code:

transpose

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Proof that H is a Parity Check Matrix

$Hy^T = 0$

$\Updownarrow$

$A^T_{i,*} \bullet y^T_{[1..k]} + y^T_{k+i} = 0$, for $1 \leq i \leq n-k$,
(where $A^T_{i,*}$ is row i of A^T).

$\Updownarrow$

$y_{[1..k]} \bullet A_{*,i} = y_{k+i}$, for $1 \leq i \leq n-k$,
(where $A_{*,i}$ is now column i of A)

$\Updownarrow$

$y_{[k+1...n]} = y_{[1..k]}A$.

$\Updownarrow$

$xG = [y_{[1..k]} \mid y_{[1..k]}A] = y$, for $x = y_{[1..k]}$

The d of linear codes

Theorem: Linear codes have distance d if every set of (d-1) columns of H are linearly independent (i.,e., cannot sum to 0), but there is a set of d columns that are linearly dependent (sum to 0).

Proof: if d-1 or fewer columns are linearly dependent, then for any codeword y, there is another codeword y', in which the bits in the positions corresponding to the columns are inverted, that both have the same syndrome, 0.

If every set of d-1 columns is linearly independent, then changing any d-1 bits in a codeword y must also change the syndrome (since the d-1 corresponding columns cannot sum to 0).

Dual Codes

For every code with

$G = I_k, A$ and $H = A^T, I_{n-k}$

we have a **dual code** with

$G = I_{n-k}, A^T$ and $H = A, I_k$

The dual of the Hamming codes are the **binary simplex codes**: $(2^r-1, r, 2^{r-1}-r)$

The dual of the extended Hamming codes are the **first-order Reed-Muller** codes.

Note that these codes are **highly redundant** and can fix many errors.

NASA Mariner:

Deep space probes from 1969-1977.

Mariner 10 shown

Used (32,6,16) Reed Muller code (r = 5)

Rate = 6/32 = .1875 (only 1 out of 5 bits are useful)

Can fix up to 7 bit errors per 32-bit word

How to find the error locations

Hy^T is called the **syndrome** (no error if 0).

In **general** we can find the error location by creating a table that maps each syndrome to a set of error locations.

Theorem: assuming $s \leq 2d-1$ every syndrome value corresponds to a unique set of error locations.

Proof: Exercise.

Table has q^{n-k} entries, each of size at most n (i.e. keep a bit vector of locations).