15-853: Algorithms in the Real World

Error Correcting Codes I
- Overview
- Hamming Codes
- Linear Codes

Mathematicians are like Frenchmen: whatever you say to them they translate into their own language and forthwith it is something entirely different.

- Goethe

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
- $\Sigma = \text{codeword alphabet}$
- $k = |m|$, $n = |c|$, $q = |\Sigma|$ $C \subseteq \Sigma^n$ (codewords)
- $\Delta(x,y) =$ number of positions $s.t. x_i \neq y_i$
- $d = \min(\Delta(x,y) : x,y \in C, x \neq y)$
- $s = \max(\Delta(c,c'))$ that the code can correct
- Code described as $(n,k,d)_t$

**Hierarchy of Codes**

- $C$ forms a linear subspace of $\Sigma^n$ of dimension $k$
- $C$ is linear and $c_0c_1c_2...c_{n-1}c_0$ is a codeword implies $c_1c_2...c_{n-1}$ is a codeword
- Bose-Chaudhuri-Hocquenghem

These are all block codes.

**Binary Codes**

- Today we will mostly be considering $\Sigma = \{0, 1\}$ 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.
- The distance between nodes on the hypercube is the Hamming distance $\Delta$. The minimum distance is $d$.
- 001 is equidistance from 000, 011 and 101.
- For $s$-bit error detection $d \geq s + 1$
- For $s$-bit error correction $d \geq 2s + 1$
Error Detection with Parity Bit

A \( (k+1,k,2) \_2 \) systematic code

**Encoding:**

\[ m_1m_2...m_k \Rightarrow m_1m_2...m_kp_{k+1} \]

where \( p_{k+1} = m_1 \oplus m_2 \oplus ... \oplus m_k \)

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 \( \Delta \) to \( k+1 \) valid codewords.

---

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

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

<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>

---

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 \( \rightarrow \) 00, 1 \( \rightarrow \) 11 (\( n=2, k=1, d=2 \))

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

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

---

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

\[ p_6 = m_6 \oplus m_4 \oplus m_{13} \oplus m_{12} \oplus m_5 \oplus m_9 \]

Localizing error to x1xx or x0xx

\[ p_4 = m_{15} \oplus m_{14} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_9 \]

Localizing error to xx1x or xx0x

\[ p_2 = m_{15} \oplus m_{14} \oplus m_{12} \oplus m_7 \oplus m_8 \oplus m_9 \]

Localizing error to xxx1 or xxx0

\[ p_1 = m_{15} \oplus m_{13} \oplus m_{12} \oplus m_7 \oplus m_8 \oplus m_3 \]

Hamming Codes: Decoding

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

After transmission, we generate:

\[ b_7 = p_8 \oplus m_{15} \oplus m_{14} \oplus m_{13} \oplus m_9 \oplus m_8 \oplus m_7 \]
\[ b_4 = p_6 \oplus m_6 \oplus m_{14} \oplus m_{13} \oplus m_9 \oplus m_8 \oplus m_7 \]
\[ b_2 = p_4 \oplus m_4 \oplus m_{14} \oplus m_{12} \oplus m_9 \oplus m_8 \oplus m_7 \]
\[ b_1 = p_4 \oplus m_4 \oplus m_{13} \oplus m_{10} \oplus m_9 \oplus m_8 \oplus m_7 \]

With no errors, these will all be zero.

With one error \( b_8 b_4 b_2 b_1 \) gives us the error location.

E.g. 0100 would tell us that \( p_4 \) is wrong, and

1100 would tell us that \( m_{12} \) 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)
- 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.

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. \( n+1 \)

\[ 2^n \geq (n+1)2^k \]
\[ n \geq k + \log_2(n+1) \]
\[ n \geq k + \lceil \log_2(n+1) \rceil \]

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

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:

\[
2^n \geq (1 + n + n(n - 1)/2)^2 \\text{\textsuperscript{k}} \\
\]

\[
n \geq k + \log_2(1 + n + n(n - 1)/2) \\text{\textsuperscript{k}} \\
\]

\[
\geq k + 2\log_2 n - 1 \\text{\textsuperscript{k}}
\]

Generally to correct $s$ errors:

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

**Linear Codes**

If $\Sigma$ is a field, then $\Sigma^n$ is a vector space

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

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

i.e. every codeword can be written as:

\[
c = a_1 v_1 + \ldots + a_k v_k \quad a_i \in \Sigma
\]

The sum of two codewords is a codeword.

**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:

```
---X---X---X---X---X---X---X---X---
```

Can we do better if we make 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**

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

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0</td>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

How can we see that $d = 3$?

For all binary linear codes, the minimum distance is equal to the least weight non-zero codeword.
**Generator and Parity Check Matrices**

**Generator Matrix:**
A $k \times n$ matrix $G$ such that: $C = \{xG \mid x \in \Sigma^k\}$
Made from stacking the basis vectors

**Parity Check Matrix:**
An $(n - k) \times n$ 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

$HG^T = 0$ since:
$0 = Hy^T = H(xG)^T = H(G^Tx^T) = (HG^T)x^T$
only true for all $x$ if $HG^T = 0$

---

**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".

\[
g = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1
\end{bmatrix}
\]

---

**Advantages of Linear Codes**

- Encoding is efficient (vector-matrix multiply)
- Error detection is efficient (vector-matrix multiply)
- Syndrome $(Hy^T)$ has error information
- Gives $2^{n-k}$ sized table for decoding
  Useful if $n-k$ is small

---

**Relationship of $G$ and $H$**

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

**Proof:**
$HG^T = A^TI_k + I_{n-k}A^T = A^T + A^T = 0$

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

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1
\end{bmatrix}
\]

\[
H = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix}
\]
The d of linear codes

**Theorem:** Linear codes have distance $d$ if every set of $(d-1)$ columns of $H$ are linearly independent, but there is a set of $d$ columns that are linearly dependent.

**Proof summary:** if $d$ columns are linearly dependent then there exist two codewords that differ in the $d$ bits corresponding to those columns that make the same contribution to the syndrome.

---

Dual Codes

For every code with

$$G = I_nA$$

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:


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 $q^n$ entries, each of size at most $n$ (i.e. keep a bit vector of locations).