296.3: Algorithms in the Real World

Error Correcting Codes II
- Cyclic Codes
- Reed-Solomon Codes

Viewing Messages as Polynomials

A (n, k, n-k+1) code:
Consider the polynomial of degree k-1
\[ p(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + a_0 \]

Message: \((a_{k-1}, \ldots, a_1, a_0)\)
Codeword: \((p(y_0), p(y_1), \ldots, p(y_{n-1}))\) for distinct \(y_0, \ldots, y_{n-1}\)

To keep the \(p(y_i)\) fixed size, we use \(y_i, a_i \in GF(p^r)\)
To make the \(y_i\) distinct, \(n < p^r\)

Unisolvence Theorem: Any subset of size \(k\) of \((p(y_1), p(y_2), \ldots, p(y_n))\) is enough to (uniquely) reconstruct \(p(x)\) using polynomial interpolation, e.g., LaGrange's Formula.

Polynomial-Based Code

A \((n, k, 2s+1)\) code:

Can detect 2s errors
Can correct s errors
Generally can correct \(\alpha\) erasures and \(\beta\) errors if \(\alpha + 2\beta \leq 2s\)

Correcting Errors

Correcting s errors:
1. Find \(k + s\) symbols that agree on a polynomial \(p(x)\).
   These must exist since originally \(k + 2s\) symbols agreed and only \(s\) are in error
2. There are no \(k + s\) symbols that agree on the wrong polynomial \(p'(x)\)
   - Any subset of \(k\) symbols will define \(p'(x)\)
   - Since at most \(s\) out of the \(k+s\) symbols are in error, \(p'(x) = p(x)\)
**A Systematic Code**

**Message:** \((m_0, m_1, \ldots, m_{k-1})\)

Find polynomial \(p(x) = a_{k-1}x^{k-1} + \ldots + a_1x + a_0\) such that
\[ p(y_i) = m_i, \quad i = 0, 1, \ldots, k-1 \]

**Codeword:** \((m_0, m_1, \ldots, m_{k-1}, p(y_{k}), p(y_{k+1}), \ldots, p(y_{n-1}))\)

This has the advantage that if we know there are no errors (e.g., all points lie on the same degree \(k-1\) polynomial), it is trivial to decode.

The version of RS used in practice uses something slightly different.

This will allow us to use the "Parity Check" ideas from linear codes (i.e., \(Hc^T = 0\)) to quickly test for errors.

---

**Reed-Solomon Codes in the Real World**

- \((204,188,17)_{256} : \text{ITU J.83(A)}^2\)
- \((128,122,7)_{256} : \text{ITU J.83(B)}\)
- \((255,223,33)_{256} : \text{Common in Practice}\)

- Note that they are all byte based (i.e., symbols are from \(GF(2^8)\)).

Decoding rate on 1.8GHz Pentium 4:
- \((255,251) = 89\text{Mbps}\)
- \((255,223) = 18\text{Mbps}\)

Dozens of companies sell hardware cores that operate 10x faster (or more):
- \((204,188,17) = 320\text{Mbps (Altera decoder)}\)

---

**Applications of Reed-Solomon Codes**

- **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, ..

Good at handling burst errors.

Other codes are better for random errors.
- e.g., Gallager codes, Turbo codes

---

**RS and “burst” errors**

Let’s compare to Hamming Codes (which are "optimal").

<table>
<thead>
<tr>
<th>Code</th>
<th>Code Bits</th>
<th>Check Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS ((255, 253, 3)_{256})</td>
<td>2040</td>
<td>16</td>
</tr>
<tr>
<td>Hamming ((2^{11}-1, 2^{11}-11-1, 3)_{256})</td>
<td>2047</td>
<td>11</td>
</tr>
</tbody>
</table>

They can both correct 1 error, but not 2 random errors.
- The Hamming code does this with fewer check bits
However, RS can fix 8 contiguous bit errors in one byte
- Much better than lower bound for 8 arbitrary errors

\[ \log\left( 1 + \frac{n}{1} + \ldots + \frac{n}{8} \right) > 8\log(n - 7) = 88 \text{ check bits} \]
**Galois Field**

$GF(2^3)$ with irreducible polynomial: $x^3 + x + 1$

$\alpha = x$ is a generator

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$x$</th>
<th>010</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^2$</td>
<td>$x^2$</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>$\alpha^3$</td>
<td>$x + 1$</td>
<td>011</td>
<td>4</td>
</tr>
<tr>
<td>$\alpha^4$</td>
<td>$x^2 + x$</td>
<td>110</td>
<td>5</td>
</tr>
<tr>
<td>$\alpha^5$</td>
<td>$x^2 + x + 1$</td>
<td>111</td>
<td>6</td>
</tr>
<tr>
<td>$\alpha^6$</td>
<td>$x^2 + 1$</td>
<td>101</td>
<td>7</td>
</tr>
<tr>
<td>$\alpha^7$</td>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
</tbody>
</table>

Will use this as an example.

---

**Discrete Fourier Transform (DFT)**

Evaluating polynomial at $n$ points via matrix multiply:

$\alpha$ is a primitive $n^{th}$ root of unity ($\alpha^n = 1$) – a generator

$T = \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\
1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{(n-1)(n-1)}
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_{k-1}
\end{pmatrix}
= T^T \begin{pmatrix}
m_0 \\
m_1 \\
m_2 \\
\vdots \\
m_k
\end{pmatrix}$

Evaluate polynomial $m_{k-1}x^{k-1} + \cdots + m_0 x + m_0$ at $n$ distinct roots of unity, $1$, $\alpha$, $\alpha^2$, ..., $\alpha^{n-1}$

Inverse DFT: $m = T^{-1}c$

---

**DFT Example**

$\alpha = x$ is 7th root of unity in $GF(2^3)/x^3 + x + 1$

(i.e., multiplicative group, which excludes additive inverse)

Recall $\alpha = "2", \alpha^2 = "3", \ldots, \alpha^7 = "1"$

$T = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
1 & \alpha^2 & \alpha^4 & \alpha^8 & \alpha^6 & \alpha^2 & 1 \\
1 & \alpha^4 & \alpha^8 & 1 & \alpha^2 & \alpha^6 & \alpha^4 \\
1 & \alpha^6 & \alpha^2 & \alpha^4 & \alpha^8 & 1 & \alpha^2 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6
\end{pmatrix}$

$= \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 2^2 & 2^3 & 2^4 & 2^5 & 2^6 \\
1 & 3 & 3^2 & 3^3 & 3^4 & 3^5 & 3^6 \\
1 & 4 & 4^2 & 4^3 & 4^4 & 4^5 & 4^6 \\
1 & 5 & 5^2 & 5^3 & 5^4 & 5^5 & 5^6 \\
1 & 6 & 6^2 & 6^3 & 6^4 & 6^5 & 6^6 \\
1 & 7 & 7^2 & 7^3 & 7^4 & 7^5 & 7^6
\end{pmatrix}$

Should be clear that $c = T \cdot (m_0, m_1, \ldots, m_k, 0, \ldots)^T$ is the same as evaluating $p(x) = m_0 + m_1 x + \cdots + m_k x^{k-1}$ at $n$ points.

---

**Decoding**

Why is it hard?

Brute Force: try $k+2s$ choose $k+s$ possibilities and solve for each.
Cyclic Codes

A linear code is cyclic if:

\[(c_0, c_1, \ldots, c_{n-1}) \in C \Rightarrow (c_{n-1}, c_0, \ldots, c_{n-2}) \in C\]

Both Hamming and Reed-Solomon codes are cyclic.

Motivation: They are more efficient to decode than general codes.

Linear Code Generator and Parity Check Matrices

View message, codeword as vectors \((m_0, m_1, \ldots, m_{k-1})\) and \((c_0, c_1, \ldots, c_{n-1})\)

**Generator Matrix:**

A \(k \times n\) matrix \(G\) such that:

\[C = \{m \cdot G \mid m \in \Sigma^k\}\]

Made from stacking the basis vectors

**Parity Check Matrix:**

A \((n - k) \times n\) matrix \(H\) such that:

\[C = \{v \in \Sigma^n \mid H \cdot v^T = 0\}\]

Codewords are the nullspace of \(H\)

These always exist for linear codes

\[H \cdot G^T = 0\]

RS Generator and Parity Check Polynomials

View message \((m_0, m_1, \ldots, m_{k-1})\) as polynomial \(m_0 + m_1x + \ldots + m_{k-1}x^{k-1}\),

codeword \((c_0, c_1, \ldots, c_{n-1})\) as polynomial \(c_0 + c_1x + \ldots + c_{n-1}x^{n-1}\)

**Generator Polynomial:**

A degree \((n-k)\) polynomial \(g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}\) such that:

\[C = \{m \cdot g \mid m \in \Sigma^k, m_0 + m_1x + \ldots + m_{k-1}x^{k-1}\}\]

such that \(g \mid x^n - 1\)

**Parity Check Polynomial:**

A degree \(k\) polynomial \(h(x) = h_0 + h_1x + \ldots + h_kx^k\) such that:

\[C = \{v \in \Sigma^n[x] \mid h \cdot v = 0 \pmod{x^n - 1}\}\]

such that \(h \mid x^n - 1\)

These always exist for linear cyclic codes

\(h \cdot g = x^n - 1\)

Poly multiplication via matrix multiplication

If \(g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}\)

We can put this generator in matrix form \((k \times n)\):

\[
G = \begin{pmatrix}
    g_0 & g_1 & \cdots & g_{k-1} & g_{n-k} \\
    0 & g_0 & \cdots & \cdots & g_{n-k-1} \\
    \vdots & \vdots & \ddots & \ddots & \vdots \\
    0 & 0 & \cdots & g_0 & \cdots \\
    \vdots & \vdots & \ddots & \ddots & \ddots \\
    \end{pmatrix}
\]

Write \(m = m_0 + m_1x + \ldots + m_{k-1}x^{k-1}\) as \((m_0, m_1, \ldots, m_{k-1})\)

Then \(c = mG\)
g generates cyclic codes

\[ g \text{ generates cyclic codes} \]

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

Codes are linear combinations of the rows.
All but last row is clearly cyclic (left shift of next row)
Right shift of last row is \( x^{n-1} \) \( \mod (x^n - 1) \) = \( g_{n-k}, \ldots, g_0 \), \( g_{n-k-1} \)
Will show \( x^{n-1} \) \( \mod (x^n - 1) \) is a linear combination of other rows.

Consider \( h = h_0 + h_1 x + \cdots + h_k x^k \)
we can put this parity check poly. in matrix form \( ((n-k) \times n) \):

\[ H = \begin{pmatrix} 0 & \cdots & 0 & h_k & \cdots & h_1 & h_0 \\ 0 & \cdots & h_k & h_{k-1} & \cdots & h_1 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ h_k & \cdots & h_1 & 0 & \cdots & 0 \end{pmatrix} \]

\( Hc^T = 0 \text{ (syndrome gives coefficients of } x^{n-1} \text{ through } x^k \text{ in } h \cdot c, \text{ which are the same as in } h \cdot c \text{ mod } x^n - 1) \)

Hamming Codes Revisited

The Hamming \((7,4,3)\), \(2\) code.

\[ g = 1 + x + x^3 \]

\[ h = x^4 + x^2 + x + 1 \]

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

\[ h \cdot g = x^7 - 1, \quad GH^T = 0 \]

The columns are not identical to the previous example Hamming code.

Factors of \( x^n - 1 \)

Intentionally left blank
Another way to write \( g \)

Let \( \alpha \) be a generator of \( GF(p^r) \).
Let \( n = p^r - 1 \) (the size of the multiplicative group).
Then we can write a generator polynomial as
\[
g(x) = (x-\alpha)(x-\alpha^2) ... (x-\alpha^{n-k}) = (x-\alpha^{n-k+1}) ... (x-\alpha^n)
\]

Lemma: \( g | x^n - 1, h | x^n - 1, gh | x^n - 1 \) (\( a | b \) means \( a \) divides \( b \))

Proof:
- \( \alpha^n = 1 \) (because of the size of the group)
  \( \Rightarrow \alpha^n - 1 = 0 \)
  \( \Rightarrow \alpha \) root of \( x^n - 1 \)
  \( \Rightarrow (x - \alpha) | x^n - 1 \)
- similarly for \( \alpha^2, \alpha^3, ..., \alpha^n \)
- therefore \( x^n - 1 \) is divisible by \( (x - \alpha)(x - \alpha^2) ... \)

Back to Reed-Solomon

Consider a generator polynomial \( g \in GF(p^r)[x] \), s.t. \( g \mid (x^n - 1) \).
Recall that \( n - k = 2s \) (the degree of \( g \) is \( n-k \), \( n-k+1 \) coefficients).

**Encode** (trick to make code systematic):
- \( m' = m \times \alpha^{2s} \) (basically shift by \( 2s \))
- \( b = m' \mod g \), \( m' = qg + b \) for some \( q \)
- \( c = m' - b = (m_{k-1}, ..., m_0, -b_{2s}, ..., -b_0) \)
- Note that \( c \) is a cyclic code based on \( g \)
  \( c = m' - b = qg \)
  (i.e., given \( m \) we found another message \( q \) such that \( qg \)
  is "systematic" for \( m \))

**Parity check:**
- \( h \cdot c = 0 \mod (x^n - 1) \)

Example RS \((7,3,5)_8\)

\( n = 7, k = 3, n-k = 2s = 4, d = 2s+1 = 5 \)
\n\[
g = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)
= x^4 + \alpha x^3 + \alpha^2 x^2 + \alpha^3 x + \alpha^4
\]
\[
h = (x - \alpha^2)(x - \alpha^3)(x - \alpha^7)
= x^3 + \alpha^2 x^2 + \alpha x + \alpha^4
\]
\[
gh = x^7 - 1
\]

Consider the message: 110 000 110
\( m = (\alpha^5, 0, \alpha^3) = \alpha^5 x^2 + \alpha^4 \)
\( m' = \alpha^4 m = \alpha^4 x^6 + \alpha^3 x^4 \)
\( = (\alpha^4 x^2 + x^2 + \alpha^3)g + (\alpha^3 x^3 + \alpha^6 x + \alpha^6) \)
\( c = (\alpha^5, 0, \alpha^3, 0, \alpha^6, \alpha^6) \)
\( = 110 000 110 011 000 101 101 \)
\( ch = 0 \mod (x^7 - 1) \)
**A useful theorem**

**Theorem:** For any \( \beta \), if \( g(\beta) = 0 \) then \( \beta^2 m(\beta) = b(\beta) \)

**Proof:**
\[
x^2m(x) = m'(x) = g(x)q(x) + b(x)
\]
\[
\beta^2m(\beta) = g(\beta)q(\beta) + b(\beta) = b(\beta)
\]

**Corollary:** \( \beta^2 m(\beta) = b(\beta) \) for \( \beta \in \{ \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2s+n-k} \} \)

**Proof:**
\[
\{ \alpha, \alpha^2, \ldots, \alpha^{2s} \} \text{ are the roots of } g \text{ by definition.}
\]

**Fixing errors**

**Theorem:** Any \( k \) symbols from \( c \) can reconstruct \( c \) and hence \( m \)

**Proof:**
We can write \( 2s \) equations involving \( m \) (\( c_{n-1}, \ldots, c_{2s} \)) and \( b \) (\( c_{2s-1}, \ldots, c_0 \)). These are
\[
\alpha^{2s} m(\alpha) = b(\alpha)
\]
\[
\alpha^{4s} m(\alpha^2) = b(\alpha^2)
\]
\[
\ldots
\]
\[
\alpha^{2s(2s)} m(\alpha^{2s}) = b(\alpha^{2s})
\]
We have at most \( 2s \) unknowns, so we can solve for them. (I'm skipping showing that the equations are linearly independent).

**Efficient Decoding**

I don't plan to go into the Reed-Solomon decoding algorithm, other than to mention the steps.

This is the hard part. CD players use this algorithm. (Can also use Euclid's algorithm.)