MSRI-UP 2009 Coding Theory Seminar, Week 3

John B. Little

Department of Mathematics and Computer Science
College of the Holy Cross

June 29 - July 1, 2009

Outline

Introduction

The Reed-Solomon codes are cyclic codes over the alphabet $\mathbb{F}_{2^r}$ with many good properties:

- Best possible d for their n and k – indeed, they are codes achieving the Singleton bound: $d = n - k + 1$,
- Good encoding method (via polynomial division – you will see this in the computer lab)
- Efficient decoding methods (Berlekamp-Massey, Euclidean Algorithm decoders)
- As a result, they are widely used in applications

Cyclic codes over $\mathbb{F}_{2^r}$

Almost all of the facts about binary cyclic codes we saw last week extend to the case of cyclic codes over the alphabet $\mathbb{F}_{2^r}$:

- (In polynomial form), the codewords are all multiples of a *generator polynomial* $g(x) \in \mathbb{F}_{2^r}[x]$ (a factor of $x^n + 1$)
- Generator and parity-check matrices can be formed as before
- The roots of the generator polynomial now determine "honest" parity-check matrices over the field $\mathbb{F}_{2^r}$.

An example

Construct $\mathbb{F}_{2^3} = \mathbb{F}_2[x]/\langle x^3 + x + 1\rangle$, so $\alpha = x$ is a primitive element.

- Let $n = 7$, so
$$x^7 + 1 = \prod_{\beta \neq 0 \in \mathbb{F}_{2^3}} (x + \beta).$$
- We can take any $g(x)$ dividing this to get a generator polynomial for a cyclic code C with $n = 7$.
- For instance, take
$$\begin{aligned} g(x) &= (x+\alpha)(x+\alpha^2)(x+\alpha^3) \\ &= x^3 + \alpha^6 x^2 + \alpha x + \alpha^6 \end{aligned}$$

Notes

Example, continued

From the expanded form of the generator polynomial, we get

$$G = \begin{pmatrix} \alpha^6 & \alpha & \alpha^6 & 1 & 0 & 0 & 0 \\ 0 & \alpha^6 & \alpha & \alpha^6 & 1 & 0 & 0 \\ 0 & 0 & \alpha^6 & \alpha & \alpha^6 & 1 & 0 \\ 0 & 0 & 0 & \alpha^6 & \alpha & \alpha^6 & 1 \end{pmatrix}$$

Notes

Example, continued

Since every codeword must have roots $\alpha, \alpha^2, \alpha^3$, we get a parity-check matrix:

$$\begin{pmatrix} 1 & 1 & 1 \\ \alpha & \alpha^2 & \alpha^3 \\ \alpha^2 & (\alpha^2)^2 & (\alpha^3)^2 \\ \vdots & \vdots & \vdots \\ \alpha^6 & (\alpha^2)^6 & (\alpha^3)^6 \end{pmatrix}$$

The code has $n = 7$, $k = 4$ over $\mathbb{F}_{2^3}$.

Notes

Reed-Solomon codes

Our example code is in fact a first example of a Reed-Solomon code.

Definition

Let α be a primitive element for the field $\mathbb{F}_{2^r}$. A **Reed-Solomon code** *$RS(2^r, \delta)$ is a cyclic code of length $n = 2^r - 1$ over $\mathbb{F}_{2^r}$ whose generator polynomial has the form*

$$g(x) = (x + \alpha^{m+1})(x + \alpha^{m+2}) \cdots (x + \alpha^{m+\delta-1})$$

for some m.

(Note – the roots of $g(x)$ are a *consecutive string* of $\delta - 1$ powers of α.)

Notes

History

- RS codes are named after their inventors (discoverers?), Irving Reed and Gustave Solomon.
- The codes were invented in 1960, when Reed and Solomon worked at MIT's Lincoln Labs in Massachusetts.
- Reed, who is still living, earned his Ph.D. at Cal Tech and later taught at USC before retiring.
- Solomon, who died in 1996, earned his Ph.D. at MIT, and consulted for many years at JPL in Pasadena.

Notes

Minimum distance of $RS(2^r, \delta)$.

- The argument with Vandermonde determinants that we used to estimate d for the BCH codes also applies here.
- From the form of H for Reed-Solomon code, H has $\delta - 1$ columns.
- Any set of $\delta - 1$ rows of H is *linearly independent* over $\mathbb{F}_{2^r}$, since (after factoring out common factors in the rows) the $(\delta - 1) \times (\delta - 1)$ submatrix is a Vandermonde matrix.
- It follows that $d = \delta$.
- Note that $n = 2^r - 1$ and $k = 2^r - \delta$. Hence $d = \delta = n - k + 1$. (Singleton bound is reached!)

Notes

A comment

- The $RS(2^r, \delta)$ codes all have $n = 2^r - 1$ exactly.
- This is somewhat restrictive for use in applications.
- We can also define *shortened* RS codes of length any $n < 2^r - 1$.
- Idea – pick any s locations in the words ($\Leftrightarrow$ a set of s nonzero elements of $\mathbb{F}_{2^r}$), take the *subcode* of $RS(2^r, \delta)$ with zeroes in those locations, and delete the zeroes.
- Leaves a code $C(s)$ with $n = 2^r - 1 - s$, $k = 2^r - \delta - s$, $d = \delta$. These codes meet the Singleton bound too!

Notes

An alternate construction of $RS(2^r, \delta)$

- The construction of $RS(2^r, \delta)$ from a particular form of generator polynomial is *not the original way* that Irving Reed and Gustave Solomon defined these codes.
- We will study that original construction next, because it gives another very nice way to understand d (maybe even clearer as motivation)
- However, it is slightly *tricky*, since it uses polynomials in $\mathbb{F}_{2^r}[x]$ in a *different way* from the way we have associated polynomials to codewords.
- This alternate construction is closely related to one of the groups of research project topics, though, so it will be valuable to understand this in detail(!)

Notes

The evaluation mapping

- Let us start with the desired *dimension* $k < 2^r$.
- Let $L_k = \mathrm{Span}\{1, t, t^2, \ldots, t^{k-1}\} \subset \mathbb{F}_{2^r}[t]$.
- We can define a code of dimension k by *evaluating* polynomials $f \in L_k$ to get the codeword entries:

$$\begin{aligned} ev : L_k &\longrightarrow \mathbb{F}_{2^r}^{2^r-1} \\ f &\longmapsto (f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{2^r-2})) \end{aligned}$$

  (where α is a primitive element).
- The image of *ev* is a linear code of dimension k – we claim that it is in fact an RS code(!)

Notes

Why is this an RS code?

- We must show that if we take the polynomial form of these codewords, then all of them have some consecutive string of powers of α as roots.
- This is easiest to see for the codewords $ev(t^i)$ for $i = 0, \ldots, k-1$.
- The polynomial form of $ev(t^i)$ is

$$1 + \alpha^i x + \alpha^{2i} x^2 + \cdots + \alpha^{(2^r-2)i} x^{2^r-2},$$

- which is the same as $p(\alpha^i x)$ for $p(v) = 1 + v + \cdots + v^{2^r-2}$.

Notes

Why is this an RS code?, continued

- Note that $(1+v)p(v) = 1 + v^{2^r-1}$.
- It follows that the roots of $p(v)$ are all the nonzero $\beta \neq 1$ in $\mathbb{F}_{2^r}$.
- Hence the roots of $p(\alpha^i x) = 0$ are all the nonzero $x \neq \alpha^{-i} = \alpha^{2^r-1-i}$ in $\mathbb{F}_{2^r}$.
- Letting $i = 0, 1, \ldots, k-1$ we see that the common roots of all the codewords of our code are

$$\alpha, \alpha^2, \ldots, \alpha^{2^r-k-1}$$

- In other words, $ev(L_k)$ is the same as the $RS(2^r, 2^r-k)$ code we saw before (with $m = 0$).

Notes

$d \Leftrightarrow$ a basic fact for polynomials

- With this alternate way to understand where the RS codewords come from, note that our determination of d just comes down to asking, how many zeroes can a nonzero polynomial in L_{k-1} have?
- The answer is clear – no more than $k-1$ roots!
- **Proof:** By division algorithm, β is a root of $f(x)$ if and only if $f(x) = (x-\beta)q(x)$ with $\deg(q(x)) = \deg(f(x)) - 1$. We then continue with $q(x)$ to see that $f(x)$ has at most $k-1$ roots. $\square$
- Moreover, *some* polynomials of degree $k-1$ have $k-1$ distinct roots.
- So the nonzero words of minimum weight in $ev(L_{k-1})$ have weight $d = 2^r - k = \delta$ from before.
- Once again, $d = n - k + 1$!

Notes

A final example

To tie everything together, let us give the two constructions of an $RS(2^4, 7)$ code.

- The generator polynomial for the RS code with $m = 0$ is $g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^6)$ ($\deg(g) = 6$)
- Since $n = 2^4 - 1 = 15$, this means that $k = 9$.
- So $RS(2^4, 7)$ can also be constructed as $ev(L_9)$ for $L_9 = \{1, t, t^2, \ldots, t^8\} \subset \mathbb{F}_{2^4}[t]$.
- Gives two different ways to produce generator matrices, either the "cyclic" generator matrix from $g(x)$, or the matrix whose ith row is

$$ev(t^i) = (1, \alpha^i, \alpha^{2i}, \ldots, \alpha^{(2^r-2)i})$$

 ($i = 0, \ldots, 8$).

Notes

Introduction

We will now develop an efficient decoding method for the $RS(2^r, \delta)$ codes. (Note – our method has much of the same algebraic background as the Berlekamp-Massey algorithm presented in the text, but it uses a *different method* to produce solutions of the *key equation* for decoding.)

Our method will rely on the (generalized) Euclidean algorithm for polynomials, an extension of the algorithm we discussed in Week 1 for finding $\gcd(f(x), g(x))$.

Notes

Background – the generalized Euclidean algorithm

- Recall that we have seen that if $d(x) = \gcd(f(x), g(x))$, then there are $A(x), B(x)$ such that $d(x) = A(x)f(x) + B(x)g(x)$.
- There is a version of the Euclidean Algorithm that computes $d(x)$ together with the $A(x), B(x)$.
- We first introduce the notation $f(x) = r_{-1}(x)$ and $g(x) = r_0(x)$ to give a uniform form for the steps in the successive divisions.
- So every line of the computation of the remainder sequence can be written as

$$r_{k-1}(x) = q_k(x)r_k(x) + r_{k+1}(x)$$

for $k = 0, 1, 2, \ldots$.

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Generalized Euclid, continued

```
Input: nonzero f(x), g(x)
Output: d(x), A(x), B(x)
r[-1] := f; r[0] := g;
A[-1] := 1; A[0] := 0;
B[-1] := 0; B[0] := 1;
k:=0;
while r[k] <> 0 do
   r[k+1] := rem(r[k-1],r[k],x);
   q[k] := quo(r[k-1],r[k],x);
   A[k+1] := A[k-1] - q[k] A[k];
   B[k+1] := B[k-1] - q[k] B[k];
   k := k+1
end do;
```

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

Notes

Note: The polynomials $A(x), B(x)$, and $d(x)$ are the final values $A[k]$, $B[k]$, and $r[k]$, respectively.

There is a nice tabular format for organizing and carrying out these calculations, shown in the following example.

Notes

Let $f(x) = x^6 + x^5 + x^3 + x^2$ and $g(x) = x^6 + x^4 + x + 1$ in $\mathbb{F}_2[x]$.

k	$r[k]$	$q[k]$	$A[k]$	$B[k]$
-1	$x^6 + x^5 + x^3 + x^2$		1	0
0	$x^6 + x^4 + x + 1$	1	0	1
1	$x^5 + x^4 + x^3 + x^2 + x + 1$	$x + 1$	1	1
2	$x^4 + x$	$x + 1$	$x + 1$	x
3	$x^3 + 1$	x	x^2	$x^2 + x + 1$

Then, $(x^2)f(x) + (x^2 + x + 1)g(x) = x^3 + 1$ as claimed.

Setup for decoding

- Assume $d = 2t + 1$. Then any t or fewer symbol-level errors in a received word are correctable.
- Let $u = \sum_{j=0}^{2^r-2} u_j x^j$ be a codeword of C.
- In $\mathbb{F}_{2^r}[x]$, u is divisible by the generator polynomial $g = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{d-1})$.
- Suppose that u is transmitted, but some errors are introduced, so that the received word is $r = u + e$ for some $e = \sum_{i \in L} e_i x^i$.
- L is called the set of *error locations*, and we assume $|L| \leq t$. The coefficients e_i are known as the *error values*.

Notes

The decoding problem

Decoding Problem: Given the received word r, determine the set of error locations L and the error values e_i for the error polynomial e with t or fewer nonzero terms (if such a polynomial exists).

Once we find e, the decoding function will return $E^{-1}(r - e)$.

Notes

The error syndromes

- The values of the polynomial form of the received word at $\alpha, \ldots, \alpha^{d-1}$ are known as the *error syndromes*.
- If $r(\alpha^j) = 0$ for all $j = 1, \ldots, d-1$, then r is divisible by g, and assuming t or fewer errors have occurred, r must be the codeword we intended to send.
- Note that for $j = 1, \ldots, d-1$,

$$s_j = r(\alpha^j) = u(\alpha^j) + e(\alpha^j) = e(\alpha^j),$$

since u is a multiple of g. Hence the s_j are the values of the error polynomial for $j = 1, \ldots, d-1$.

Notes

The syndrome polynomial and series

- The syndromes may be used as the coefficients in a polynomial

$$\Sigma(x) = \sum_{j=1}^{d-1} s_j x^{j-1},$$

called the *syndrome polynomial* for the received word r.
- The degree of Σ is $d-2$ or less.
- By extending the definition of $s_j = e(\alpha^j)$ to *all* exponents j we can also consider the formal power series

$$\widehat{\Sigma}(x) = \sum_{j=1}^{\infty} s_j x^{j-1}.$$

Notes

Preparation for key equation

- Suppose we knew the error polynomial e. Then, $s_j = \sum_{i \in L} e_i (\alpha^j)^i = \sum_{i \in L} e_i (\alpha^i)^j$.
- By algebraic manipulation, $\widehat{\Sigma}(x)$ can be written as

$$\begin{aligned}\widehat{\Sigma}(x) = \sum_{j=1}^{\infty} s_j x^{j-1} &= \sum_{i \in L} e_i \left(\sum_{j=1}^{\infty} (\alpha^i)^j x^{j-1} \right) \\ &= \sum_{i \in L} \frac{e_i \alpha^i}{(1-\alpha^i x)} \\ &= \frac{\Omega(x)}{\Lambda(x)},\end{aligned}$$

where $\Lambda(x) = \prod_{i \in L} (1 - \alpha^i x)$ and $\Omega(x) = \sum_{i \in L} e_i \alpha^i \prod_{j \in L, j \neq i} (1 - \alpha^j x)$.

Notes

Error locator and error evaluator

- The roots of Λ are precisely the α^{-i} for $i \in L$.
- Since the error locations can be determined easily from these roots, Λ is called the *error locator polynomial*.
- $\deg \Omega \leq \deg \Lambda - 1$.
- In addition, if $i \in L$,

$$\Omega(\alpha^{-i}) = e_i \alpha^i \prod_{j \in L, j \neq i} (1 - \alpha^j \alpha^{-i}) \neq 0.$$

- Hence Ω and Λ must be *relatively prime*.

Notes

The “tail” of the series

- Similarly, if we consider the “tail” of the series $\widehat{\Sigma}$,

$$\begin{aligned}\widehat{\Sigma}(x) - \Sigma(x) &= \sum_{j=d}^{\infty} \left(\sum_{i \in L} e_i (\alpha^i)^j \right) x^{j-1} \\ &= x^{d-1} \cdot \frac{\Gamma(x)}{\Lambda(x)},\end{aligned}$$

  where $\Gamma(x) = \sum_{i \in L} e_i \alpha^{id} \prod_{j \in L; j \neq i} (1 - \alpha^j x)$.
- The degree of Γ is also at most $\deg \Lambda - 1$.

Notes

The key equation

Combining the above, and writing $d - 1 = 2t$ we obtain the relation

$$\boxed{\Omega(x) = \Lambda(x)\Sigma(x) + x^{2t}\Gamma(x),}$$

called the *key equation* for decoding.

Decoding will be accomplished if we can *solve* the key equation for the unknowns $\Lambda(x)$, $\Omega(x)$, $\Gamma(x)$ using the known information from $\Sigma(x)$, because of the following theorem.

Notes

Decoding theorem

Theorem

Suppose that t or fewer errors occur in the received word r, and let Σ be the corresponding syndrome polynomial. Up to a constant multiple, there exists a unique solution $(\Omega, \Lambda, \Gamma)$ of the key equation that satisfies the degree conditions

$$\deg \Lambda \leq t$$
$$\deg \Omega < \deg \Lambda,$$

and for which Ω and Λ are relatively prime.

Notes

The decoding process, given a solution of the key equation

- If we can solve the key equation for $\Lambda(x), \Omega(x)$ ($\Gamma(x)$ is not actually used from this point on), then we solve $\Lambda(x) = 0$ to find the error locations. (This can be done by the "brute-force" method of searching through all nonzero elements of the field to find the roots.)
- Then the error values e_i can be determined from the equation $\Omega(x) = \sum_{i \in L} e_i \alpha^i \prod_{j \in L, j \neq i}(1 - \alpha^j x)$. (Called the "Forney formula.")

Notes

Solving the key equation

The process of solving the key equation consists of exactly the same steps as the generalized Euclidean Algorithm for the polynomials $f(x) = x^{2t}$ and $g(x) = \Sigma(x)$, *except* that we *stop* the first time we find a remainder r_k with $\deg r_k < t$ (this corresponds to an equation

$$\Gamma(x)x^{2t} + \Lambda(x)\Sigma(x) = \Omega(x)$$

with $\deg \Omega(x) < t$).

There is a Maple implementation of this procedure in the class CTP.map file.

Notes

A decoding example

- Use the $RS(8, 5)$ code ($t = 2$), with $\mathbb{F}_8$ constructed using $h(x) = x^3 + x + 1$ (α a root of this).
- The codeword $u = ev(1) = (1, 1, 1, 1, 1, 1, 1)$ is sent, but $r = (1, \alpha, 1, 1, 1, 1, \alpha^2 + 1)$.
- The first step is to compute the syndromes and the corresponding syndrome polynomial $\Sigma(x)$.
- For instance,

$$\begin{aligned} s_1 &= r(\alpha) = 1 + \alpha^2 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + \alpha^6(\alpha^2 + 1) \\ &= 1 + (\alpha + 1) + (\alpha^2 + \alpha) + (\alpha^2 + \alpha + 1) + (\alpha^2 + \alpha + 1) \\ &= \alpha^2 \end{aligned}$$

Notes

Decoding example, continued

- Similarly, $s_2 = \alpha^4$, $s_3 = 0$, $s_4 = \alpha^4$,
- Hence $\Sigma(x) = \alpha^2 + \alpha^4 x + \alpha^4 x^3$.
- We now begin the Euclidean algorithm to find the gcd of $x^{2t} = x^4$ and $\Sigma(x)$, keeping track of the remainders r_k, and A_k, B_k.
- In the first division: $x^4 = (\alpha^3 x) \cdot \Sigma(x) + (x^2 + \alpha^5 x)$, so $q_0 = \alpha^3 x$ and $r_1 = x^2 + \alpha^5 x$. Hence $A_1 = A_{-1} - q_0 A_0 = 1$ and $B_1 = B_{-1} - q_0 B_0 = \alpha^3 x$.

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Decoding example, continued

- Continue in the same way (only one more step is needed in the while loop in in this small example).
- We obtain the results collected in the table on the next slide.

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

Euclidean algorithm results

k	r_k	q_k	A_k	B_k
-1	x^4		1	0
0	$\alpha^4 x^3 + \alpha^4 x + \alpha^2$	$\alpha^3 x$	0	1
1	$x^2 + \alpha^5 x$	$\alpha^4 x + \alpha^2$	1	$\alpha^3 x$
2	$\alpha^5 x + \alpha^2$		$\alpha^4 x + \alpha^2$	$x^2 + \alpha^5 x + 1$

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

The error locations

- We stop here since $\deg(r_2) = 1 < 2 = t$. The next step is to find the roots of

$$B_2(x) = \Lambda(x) = x^2 + \alpha^5 x + 1 = 0.$$

- This can be done either by exhaustive search, or by factoring:

$$x^2 + \alpha^5 x + 1 = (1 + \alpha x)(1 + \alpha^6 x),$$

  so the roots are $x = \alpha^6, \alpha$.
- But by the definition of $\Lambda(x)$, the locations of the errors are found from the inverses: $\alpha = \alpha^{-6}$ and $\alpha^6 = \alpha^{-1}$, so the errors occurred in locations 1 and 6.

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes

The error values

- Finally we use the Forney Formula to determine the error values e_1 and e_6.
- With $x = \alpha = \alpha^{-6}$,

$$\Omega(\alpha^{-1}) = \alpha e_1 \chi_1(\alpha^6) = \alpha e_1(1 - \alpha^5) = \alpha^5 e_1.$$

- Since $\Omega(x) = \alpha^5 x + \alpha^2$, it follows that $\Omega(\alpha^{-1}) = \alpha$, and so $e_1 = \alpha^3$.
- Similarly, $e_6 = \alpha^2$.
- Hence $e(x) = \alpha^3 x + \alpha^2 x^6$ and $r(x) + e(x) = 1 + x + x^2 + \cdots + x^6$, which finishes the decoding.

Reed-Solomon Codes The Euclidean Algorithm (Sugiyama) Decoder

Notes