6. Decoding Generalized Reed-Solomon Codes

Decoding Generalized Reed-Solomon Codes

- We consider $\mathcal{C}_{\mathrm{GRS}}$ over $\mathbb{F}_q$ with PCM

$$H_{\mathrm{GRS}} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^\ell & \alpha_2^\ell & \ldots & \alpha_n^\ell \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \ldots & \alpha_n^{r-1} \end{pmatrix} \begin{pmatrix} v_1 & & & 0 \\ & v_2 & & \\ 0 & & \ddots & \\ & & & v_n \end{pmatrix}$$

  with $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_q^*$ distinct, and $v_1, v_2, \ldots, v_n \in \mathbb{F}_q^*$ (recall that $r = n - k = d - 1$).

- Codeword $\mathbf{c}$ transmitted, word $\mathbf{y}$ received, with error vector

$$\mathbf{e} = (e_1 \; e_2 \; \ldots \; e_n) = \mathbf{y} - \mathbf{c}\,.$$

- $J = \{\kappa \; : \; e_\kappa \neq 0\}$ set of *error locations*.
- We describe an algorithm that correctly decodes $\mathbf{y}$ to $\mathbf{c}$, under the assumption $|J| \leq \frac{1}{2}(d-1)$.

Syndrome Computation

- First step of the decoding algorithm: *syndrome computation*

$$\mathbf{S} = \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{r-1} \end{pmatrix} = H_{\mathrm{GRS}}\mathbf{y}^T = H_{\mathrm{GRS}}\mathbf{e}^T$$

ℓ th row of H_{GRS}:
$\left[v_1\alpha_1^\ell,\ v_2\alpha_2^\ell,\ \ldots,\ v_n\alpha_n^\ell\right]$

$$S_\ell = \sum_{j=1}^{n} y_j v_j \alpha_j^\ell = \sum_{j=1}^{n} e_j v_j \alpha_j^\ell = \sum_{j\in J} e_j v_j \alpha_j^\ell\,, \quad \ell = 0, 1, \ldots, r-1\,.$$

Example: For conventional RS codes, we have $\alpha_j = \alpha^{j-1}$ and $v_j = \alpha^{b(j-1)}$, so

$$S_\ell = \sum_{j=1}^{n} y_j \alpha^{(j-1)(b+\ell)} = y(\alpha^{b+\ell})\,, \quad \ell = 0, 1, \ldots, r-1$$

sum

(where $y(x) = \sum_{j=1}^{n} y_j x^{j-1}$; recall $\mathbf{c} \in \mathcal{C}_{\mathrm{RS}} \Leftrightarrow c(\alpha^{b+\ell}) = 0,\ \ell = 0, 1, \ldots r-1$).

- *Syndrome polynomial*:

$$S(x) = \sum_{\ell=0}^{r-1} S_\ell x^\ell = \sum_{\ell=0}^{r-1} x^\ell \sum_{j\in J} e_j v_j \alpha_j^\ell = \sum_{j\in J} e_j v_j \sum_{\ell=0}^{r-1} (\alpha_j x)^\ell\,.$$

A Congruence for the Syndrome Polynomial

$$S(x) = \sum_{j \in J} e_j v_j \sum_{\ell=0}^{r-1} (\alpha_j x)^\ell .$$

$\boxed{(1-z)\sum_{\ell=0}^{r-1} z^\ell = 1 - z^r}$

any field

- We have

$$(1-\alpha_j x)\sum_{\ell=0}^{r-1}(\alpha_j x)^\ell = 1 - (\alpha_j x)^r \equiv 1 \pmod{x^r} .$$

Therefore, we can write

$$\sum_{\ell=0}^{r-1}(\alpha_j x)^\ell \equiv \frac{1}{1-\alpha_j x} \pmod{x^r}$$

$$\Longrightarrow \qquad \boxed{S(x) \equiv \sum_{j \in J} \frac{e_j v_j}{1-\alpha_j x} \pmod{x^r}}$$

$\left(\sum_{\text{empty}} \square \triangleq 0\right)$

More Auxiliary Polynomials

- *Error locator polynomial (ELP)*

$$\Lambda(x) = \prod_{j \in J} (1 - \alpha_j x) \qquad \left(\prod_{\text{empty}} \square \triangleq 1, \text{ so } \Lambda(x) \not\equiv 0 \right)$$

- *Error evaluator polynomial (EEP)*

$$\Gamma(x) = \sum_{j \in J} e_j v_j \prod_{m \in J \setminus \{j\}} (1 - \alpha_m x)$$

- $\Lambda(\alpha_\kappa^{-1}) = 0 \quad \Longleftrightarrow \quad \kappa \in J$ *roots of ELP point to error locations*
- $\Gamma(\alpha_\kappa^{-1}) = e_\kappa v_\kappa \prod_{m \in J \setminus \{\kappa\}} (1 - \alpha_m \alpha_\kappa^{-1}) \neq 0, \quad \kappa \in J$

$$\Longrightarrow \qquad \boxed{\gcd(\Lambda(x), \Gamma(x)) = 1}$$

- The degrees of ELP and EEP satisfy

$$\deg \Lambda = |J| \quad \text{and} \quad \deg \Gamma < |J|$$

Of course, we don't know $\Lambda(x)$, $\Gamma(x)$: our goal is to find them

Key Equation of GRS Decoding

Since $|J| \leq \frac{1}{2}(d-1)$, from $\deg \Lambda = |J|$, $\deg \Gamma < |J|$ we get

(1) $\boxed{\deg \Lambda \leq \frac{1}{2}(d-1)}$ and (2) $\boxed{\deg \Gamma < \frac{1}{2}(d-1)}$

The ELP and the EEP are related by

$$\Gamma(x) = \sum_{j \in J} e_j v_j \prod_{m \in J \setminus \{j\}} (1 - \alpha_m x) = \sum_{j \in J} e_j v_j \frac{\Lambda(x)}{1-\alpha_j x} = \Lambda(x) \sum_{j \in J} \frac{e_j v_j}{1-\alpha_j x}$$

$S(x) \mod x^{d-1}$
(recall $d-1 = r$)

$\Longrightarrow$ (3) $\boxed{\Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}}$

(1)+(2)+(3): *key equation of GRS decoding*

We have $S(x)$, and we know d. We want to solve for $\Lambda(x)$ and $\Gamma(x)$ satisfying (1)+(2)+(3) (and hope the solution is unique).

Key Equation of GRS Decoding (cont.)

(1) $\boxed{\deg \Lambda \le \frac{1}{2}(d-1)}$ (2) $\boxed{\deg \Gamma < \frac{1}{2}(d-1)}$

(3) $\boxed{\Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}}$

Let $\tau = \lfloor \frac{d-1}{2} \rfloor$. Write

$$\Lambda(x) = \sum_{i=0}^{\tau} \lambda_i x^i,$$

$$\Gamma(x) = \sum_{h=0}^{\tau-1} \gamma_h x^h,$$

and recall

$$S(x) = \sum_{\ell=0}^{d-2} S_\ell x^\ell.$$

Key Equation of GRS Decoding (cont.)

(1) $\boxed{\deg \Lambda \le \frac{1}{2}(d-1)}$ (2) $\boxed{\deg \Gamma < \frac{1}{2}(d-1)}$

(3) $\boxed{\Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}}$

Writing (3) explicitly (with $\tau = \lfloor \frac{d-1}{2} \rfloor$),

$$
\begin{aligned}
\gamma_0 &= S_0\lambda_0 \\
\gamma_1 &= S_1\lambda_0 + S_0\lambda_1 \\
&\vdots \\
\gamma_{\tau-1} &= S_{\tau-1}\lambda_0 + S_{\tau-2}\lambda_1 + \cdots + S_0\lambda_{\tau-1} \\
0 &= S_\tau\lambda_0 + S_{\tau-1}\lambda_1 + \cdots + S_0\lambda_\tau \\
0 &= S_{\tau+1}\lambda_0 + S_\tau\lambda_1 + \cdots + S_1\lambda_\tau \\
&\vdots \\
0 &= S_{d-2}\lambda_0 + S_{d-3}\lambda_1 + \cdots + S_{d-2-\tau}\lambda_\tau
\end{aligned}
$$

Key Equation of GRS Decoding (cont.)

(1) $\boxed{\deg \Lambda \le \frac{1}{2}(d-1)}$ (2) $\boxed{\deg \Gamma < \frac{1}{2}(d-1)}$

(3) $\boxed{\Lambda(x)S(x) \equiv \Gamma(x) \pmod{x^{d-1}}}$

In matrix form,

$$
\begin{array}{cc}
d-1 & \left(\begin{array}{ccccc}
S_0 & 0 & 0 & \cdots & 0 \\
S_1 & S_0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
S_{\tau-1} & S_{\tau-2} & \cdots & S_0 & 0 \\
\hline
S_\tau & S_{\tau-1} & \cdots & S_1 & S_0 \\
S_{\tau+1} & S_\tau & \cdots & S_2 & S_1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
S_{d-2} & S_{d-3} & \cdots & S_{d-1-\tau} & S_{d-2-\tau}
\end{array}\right) \\
 & \tau+1
\end{array}
\begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_\tau \end{pmatrix}
=
\left(\begin{array}{c} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{\tau-1} \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right)
\qquad (**)
$$

- $(**)$ is a set of $r = d-1$ linear equations in the coefficients $\{\lambda_i\}$, $\{\gamma_h\}$.
- Let $\lambda(x), \gamma(x)$ be generic solutions to $(**)$ (of which $\Lambda(x), \Gamma(x)$ is one instance).
- The last $d-1-\tau = \lceil \frac{1}{2}(d-1) \rceil$ equations depend only on the $\{\lambda_i\}$ $\Longrightarrow$ we can solve for $\lambda(x)$ from this system of homogeneous equations.

Uniqueness of solution

Proposition

A solution $(\lambda(x), \gamma(x))$ *to* $(**)$, *with* $\lambda(x) \neq 0$ *of minimal degree is unique up to multiplication by a scalar* $c \in \mathbb{F}_q$.
It is also the unique solution for which $\gcd(\lambda(x), \gamma(x)) = 1$.

- Solving for the $\{\lambda_i\}$ gives us the coefficients of $\Lambda(x)$ (up to scalar multiplication). We can find the roots $\{\alpha_j\}$, then solve *linear* equations for the $\{e_j\}$.
- A straightforward solution, e.g. by Gaussian elimination, leads to an $O(d^3)$ algorithm—we'll present an $O(d^2)$ one.

The Extended Euclidean Algorithm for polynomials

Given $a(x), b(x)$ over a field $\mathbb{F}$, with $a(x) \neq 0$ and $\deg a > \deg b$, the algorithm computes sequences of
remainders $r_i(x)$, *quotients* $q_i(x)$, and *auxiliary polynomials* $s_i(x)$, $t_i(x)$

$$
\boxed{
\begin{array}{l}
r_{-1}(x) \leftarrow a(x);\;\; r_0(x) \leftarrow b(x); \\
s_{-1}(x) \leftarrow 1;\;\; s_0(x) \leftarrow 0; \\
t_{-1}(x) \leftarrow 0;\;\; t_0(x) \leftarrow 1; \\
\text{for } (i \leftarrow 1;\; r_{i-1}(x) \neq 0;\; i{+}{+})\ \{ \\
\quad q_i(x) \leftarrow r_{i-2}(x) \text{ div } r_{i-1}(x); \\
\quad r_i(x) \leftarrow r_{i-2}(x) - q_i(x)\, r_{i-1}(x); \\
\quad s_i(x) \leftarrow s_{i-2}(x) - q_i(x)\, s_{i-1}(x); \\
\quad t_i(x) \leftarrow t_{i-2}(x) - q_i(x)\, t_{i-1}(x); \\
\}
\end{array}
}
$$

- Let $\nu =$ largest i such that $r_i \neq 0$. Then, $r_\nu(x) = \gcd(a(x), b(x))$.
- We also know that $s_\nu(x)a(x) + t_\nu(x)b(x) = \gcd(a(x), b(x))$ (often used to compute modular inverses).

Properties of the Euclidean Algorithm Sequences

Proposition (E1)

The following relations hold:

$$\textit{(i)}\ \text{For } i = -1, 0, \ldots, \nu+1: \qquad s_i(x)a(x) + t_i(x)b(x) = r_i(x)$$

$$\textit{(ii)}\ \text{For } i = 0, 1, \ldots, \nu+1: \qquad \deg t_i + \deg r_{i-1} = \deg a$$

Proof. By induction on i.□

Proposition (E2)

Suppose that $t(x), r(x) \in \mathbb{F}[x] \setminus \{0\}$ *satisfy the following conditions:*

(C1) $\gcd(t(x), r(x)) = 1$

(C2) $\deg t + \deg r < \deg a$

(C3) $t(x)b(x) \equiv r(x) \pmod{a(x)}$

Then, for some $h \in \{0, 1, \ldots, \nu+1\}$ *and a constant* $c \in \mathbb{F}$, *we have*

$$t(x) = c \cdot t_h(x) \quad \textit{and} \quad r(x) = c \cdot r_h(x)\ .$$

Proof. Standard polynomial manipulations, Proposition (E1), and recalling that the sequence $\deg r_i$ is strictly decreasing. □

Solving the Key Equation

- Apply the Euclidean algorithm with $a(x) = x^{d-1}$ and $b(x) = S(x)$.
  - Let $\Lambda(x)$ and $\Gamma(x)$ play the roles of $t(x)$ and $r(x)$, respectively, in Proposition (E2). *The definitions of Λ and Γ, and the key equation, guarantee that conditions (C1)–(C3) are satisfied.*

    (C1) $\gcd(t(x), r(x)) = \gcd(\Lambda(x), \Gamma(x)) = 1$

    (C2) $\deg t + \deg r = \deg \Lambda + \deg \Gamma < \deg a = d - 1$

    (C3) $t(x)b(x) \equiv r(x) \bmod a(x) \Leftrightarrow \Lambda(x)S(x) \equiv \Gamma(x) \bmod x^{d-1}$
  - By Proposition (E2), we have $\Lambda(x) = c \cdot t_h(x)$ and $\Gamma(x) = c \cdot r_h(x)$ for some index h and scalar constant c. *How do we find the index h?*

Theorem

The solution to the key equation is unique up to a scalar constant. It is obtained with the Euclidean algorithm, by stopping at the unique index h such that

$$\deg r_h < \tfrac{1}{2}(d-1) \leq \deg r_{h-1}$$

Proof. Such an h exists because r_i is strictly decreasing. The degree properties (1), (2) follow from the definition of h, and Prop. (E1). □

Finding the Error Values

- *Formal derivatives* in finite fields: $\left[\sum_{i=0}^{s} a_i x^i\right]' = \sum_{i=1}^{s} i a_i x^{i-1}$
  $(a(x)b(x))' = a'(x)b(x) + a(x)b'(x)$ (not surprising)
- For the ELP, we have

$$\Lambda(x) = \prod_{j\in J}(1-\alpha_j x) \quad \Longrightarrow \quad \Lambda'(x) = \sum_{j\in J}(-\alpha_j) \prod_{m\in J\setminus\{j\}} (1-\alpha_m x)\,,$$

and, for $\kappa \in J$,

$$\Lambda'(\alpha_\kappa^{-1}) = -\alpha_\kappa \prod_{m\in J\setminus\{\kappa\}} (1-\alpha_m \alpha_\kappa^{-1})\,,$$

$$\Gamma(\alpha_\kappa^{-1}) = e_\kappa v_\kappa \prod_{m\in J\setminus\{\kappa\}} (1-\alpha_m \alpha_\kappa^{-1})$$

- Therefore, for all error locations $\kappa \in J$, we obtain

$$\boxed{e_\kappa = -\frac{\alpha_\kappa}{v_\kappa} \cdot \frac{\Gamma(\alpha_\kappa^{-1})}{\Lambda'(\alpha_\kappa^{-1})}}$$

Forney's algorithm for error values

Summary of GRS Decoding

Input: received word $(y_1\ y_2\ \ldots\ y_n) \in \mathbb{F}_q^n$.
Output: error vector $(e_1\ e_2\ \ldots\ e_n) \in \mathbb{F}_q^n$.

1. *Syndrome computation:* Compute the polynomial $S(x) = \sum_{\ell=0}^{d-2} S_\ell x^\ell$ by
$$S_\ell = \sum_{j=1}^{n} y_j v_j \alpha_j^\ell\ , \quad \ell = 0, 1, \ldots, d-2\ .$$

2. *Solving the key equation:* Apply Euclid's algorithm to $a(x) \leftarrow x^{d-1}$ and $b(x) \leftarrow S(x)$ to produce $\Lambda(x) \leftarrow t_h(x)$ and $\Gamma(x) \leftarrow r_h(x)$, where h is the smallest index i for which $\deg r_i < \frac{1}{2}(d-1)$.

3. *Root search and Forney's algorithm:* Compute the error locations and values. For $j = 1, 2, \ldots, n$:
$$e_j = \begin{cases} -\dfrac{\alpha_j}{v_j} \cdot \dfrac{\Gamma(\alpha_j^{-1})}{\Lambda'(\alpha_j^{-1})} & \text{if } \Lambda(\alpha_j^{-1}) = 0 \\ 0 & \text{otherwise} \end{cases} .$$

Complexity: 1. $O(dn)$ 2. $O\big((|J|+1)\, d\big)$ 3. $O\big((|J|+1)n\big)$

Decoding Failures and Decoding Errors

- The GRS decoding algorithm assumes several properties of the objects it constructs, derived from the initial assumption that $|J| \leq \frac{1}{2}(d-1)$.
- If the initial assumption is not true, then some of the derived properties may not hold. When this is detected, we say there is a *decoding failure*: we know errors have occurred, but we cannot correct them.
- Properties to check (assuming $\mathbf{S} \neq \mathbf{0}$):
  - $\deg \Gamma < \deg \Lambda \leq \frac{1}{2}(d-1)$.
  - The number of distinct roots of Λ in the set $\{\alpha_i^{-1} :\ 1 \leq i \leq n\}$ is equal to its degree.
  - If $\Lambda(\alpha_i^{-1}) = 0$, then $\Gamma(\alpha_i^{-1}) \neq 0$ (error values are nonzero).
- The ultimate test for decoding correctness is to check the syndrome of the corrected "codeword" $\tilde{\mathbf{c}} = \mathbf{y} - \mathbf{e}$: $H_{\mathrm{GRS}}\tilde{\mathbf{c}}^T = \mathbf{0}$ (after also verifying that $\mathrm{wt}(\mathbf{e}) \leq \frac{1}{2}(d-1)$). This has a complexity cost.
- There will be cases where $|J| > \frac{1}{2}(d-1)$ but all the other assumptions hold. In those cases, the decoder will proceed normally, and will output the wrong codeword. This situation is referred to as a *decoding error*. *Decoding failures can (and should) be detected. Decoding errors cannot.*

Decoding Errors and Erasures

- Assume a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is sent through an errors/erasures channel, and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is received, $y_i \in \mathbb{F} \cup \{\,?\,\}$.
- Define the set of *erasure locations* as $K = \{j : y_j \ = ?\}$, and the set of *error locations* as $J = \{j : j \notin K, y_j \neq c_j\}$.
  The set K is known to the decoder. As before, the set J is not.
- Recall: an error/erasures pattern is correctable iff $2|J| + |K| \leq d{-}1$.
- We modify the GRS decoding algorithm to handle *errors and erasures*.
  - Syndrome computation: for $j \in K$, set $y_j = 0$ (no $?$ in the computation).
  - The syndrome polynomial $S(x)$ and error locator polynomial $\Lambda(x)$ are defined as before. We also define the *erasure locator polynomial*

$$M(x) = \prod_{j \in K} (1 - \alpha_j x)\,.$$

  - The definition of the error evaluator polynomial is modified as

$$\Gamma(x) = \sum_{j \in K \cup J} e_j v_j \prod_{m \in (K \cup J) \setminus \{j\}} (1 - \alpha_m x)\,.$$

  $S(x)$ and $M(x)$ are *known to the decoder*.

Decoding Errors and Erasures

With $S(x)$ and $M(x)$ at hand, the algorithm proceeds as follows:

1. Let $\rho = |K|$. If $\rho > d - 1$, stop. The error pattern is *uncorrectable*.
2. Compute a *modified syndrome polynomial*
$$\tilde{S}(x) = M(x)S(x) \bmod x^{d-1} .$$
3. Run the extended Euclidean algorithm starting with $a(x) = x^{d-1}$ and $b(x) = \tilde{S}(x)$, keeping track of the polynomial sequences r_h, t_h.
4. Stop at the unique index h such that
$$\deg r_h < \tfrac{1}{2}(d + \rho - 1) \leq \deg r_{h-1} .$$
Then, $\Lambda(x) = c \cdot t_h(x)$ and $\Gamma(x) = c \cdot r_h(x)$.
5. Compute the *errors and erasures locator polynomial*
$$\tilde{\Lambda}(x) = M(x)\Lambda(x) .$$
We use $\tilde{\Lambda}(x)$ in lieu of $\Lambda(x)$ for the rest of the computation.
6. Run the Chien search with $\tilde{\Lambda}(x)$, and use Forney's formula with $\tilde{\Lambda}(x)$ and $\Gamma(x)$ to find the error and erasure locations and values. Notice that for erased locations, it is possible to get an "error value" of zero, as the erased location might have had an original value of zero.

Other Decoding Algorithms

Many decoding algorithms and variants have been developed over the years. We mention a few of the most important ones.

- *Berlekamp algorithm* [1967] (also referred to as *Berlekamp-Massey* due to a clearer description and improvements by Massey [1969]): first efficient solution of the key equation, using Newton's identities and solving for shortest recurrence that generates the syndrome sequence. Complexity comparable to the Euclidean algorithm.
- *Welch-Berlekamp* [1986]: Solves key equation starting from *remainder syndrome* $y(x) \pmod{g(x)}$, without computing power sums. Akin to continued fractions and Padé approximations.
- *List decoding*: Decodes beyond $\tau = \lfloor \frac{1}{2}(d-1) \rfloor$ errors, producing a list of candidate decoded codewords. Very often, the coset leader is unique even beyond τ. Dates back to the '50s, but has gotten recent focus due to elegant and efficient algorithms by Sudan ['97] , Guruswami-Sudan ['99] and others.
- *Soft decoding*: Information on the *reliability* of the symbols is available. Can lead to significant gains in decoding performance.