Encoding and Decoding Reed-Solomon Codes

John McCall

March 4, 2014

1 Definitions

Before we delve into encoding and decoding a Reed-Solomon Code, there are several terms that we need to define. First, a *generator matrix* of an $[n, k]$ linear code C over a field F is a $k \times n$ matrix G with C equal to the row space of G.

The *dual code* of a code C, denoted $C^{\perp}$ is the code

$$C^{\perp} = \{\mathbf{x} \in F^n \mid \mathbf{x} \cdot \mathbf{c} = 0, \text{ for all } c \in C\}$$

where $\mathbf{x} \cdot \mathbf{c}$ is the usual dot product. If C is a linear code, then $C^{\perp\perp} = C$. Dual codes are useful because they possess the property that if G is a generator matrix for C then $\mathbf{x}$ is in C if and only if $G\mathbf{x}^{\mathsf{T}} = \mathbf{0}$.

The generator matrix H for the dual code $C^{\perp}$ of linear code C is called a *check matrix* for C. Since $C^{\perp\perp} = C$, we can use the check matrix H for C to define C as:

$$C = \{\mathbf{x} \mid H\mathbf{x}^{\mathsf{T}} = \mathbf{0}\}.$$

Often a code is defined using a check matrix. [Insert example here?]

2 Reed-Solomon Basics

Let F be a field. Choose nonzero elements $v_1, ..., v_n \in F$ and distinct elements $\alpha_1, ..., \alpha_n \in F$. Set $\mathbf{v} = (v_1, ..., v_n)$ and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$ For $0 \leq k \leq n$ *generalized Reed-Solomon codes* are defined as:

$$\text{GRS}_{n,k}(\boldsymbol{\alpha}, \mathbf{v}) = \{(v_1 f(\alpha_1), v_2 f(\alpha_2), ..., v_n f(\alpha_n)) \mid f(x) \in F[x]_k\}.$$

Here $F[x]_k$ is the set of polynomials in $F[x]$ with degree less than k. If $f(x)$ is a polynomial, then $\mathbf{f}$ is its associated codeword. This codeword is also dependent on $\boldsymbol{\alpha}$ and $\mathbf{v}$. We can write

$$\mathbf{ev}_{\boldsymbol{\alpha},\mathbf{v}}(f(x)) = (v_1 f(\alpha_1), v_2 f(\alpha_2), ..., v_n f(\alpha_n)),$$

where $f = \mathbf{ev}_{\boldsymbol{\alpha},\mathbf{v}}(f(x))$ when the polynomial $f(x)$ is evaluated at $\boldsymbol{\alpha}$ and scaled by $\mathbf{v}$.

The distance between two codewords is defined as the number of symbols in which the sequences differ, in other words it is the Hamming distance. The minimum distance between two codewords for GRS codes is

$$\mathrm{d}_{\min} = n - k + 1.$$

A key concept is that any codeword which has up to k entries equal to 0 corresponds to a polynomial of degree less than k whose values matching the 0 polynomial in k points must be the 0 polynomial. This is true since any polynomial of degree less than k is uniquely determined by its values at k distinct points. Which means that for any n-tuple $\mathbf{f}$, we can reconstruct the polynomial $f(x)$ of degree less than n such that $\mathbf{f} = \mathbf{ev}_{\boldsymbol{\alpha},\mathbf{v}}(f(x))$. Let

$$L(x) = \prod_{i=1}^{n}(x - \alpha_i)$$

and

$$L_i(x) = L(x)/(x - \alpha_i) = \prod_{j\neq i}(x - \alpha_j).$$

Both $L(x)$ and $L_i(x)$ are *monic* polynomials of degrees n and $n-1$, respectively. A polynomial is monic if the leading coefficient is equal to 1. Since the i^{th} coordinate of vector $\mathbf{f}$ is $v_i f(\alpha_i)$ we can use Lagrange interpolation [insert reference to Lagrange interpolation here] to calculate

$$f(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(\alpha_i)} f(\alpha_i).$$

The polynomial $f(x)$ has degree less than k, while the interpolation polynomial of the righthand side has degree of $n-1$. The solution to this problem allows us to calculate the dual of a GRS code more easily. Hall gives a theorem in Chapter 5 of *Notes on Coding Theory* stating that:

$$\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \mathbf{v})^{\perp} = \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, \mathbf{u}),$$

where $\mathbf{u} = (u_1, ..., u_n)$ with $u_i^{-1} = v_i \prod_{j\neq i}(\alpha_i - \alpha_j)$.

To verify that $\mathbf{f}$ is a codeword in $C = \mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$ it is not necessary to compare it to every $\mathbf{g}$ of $C^{\perp} = \mathrm{GRS}_{n,n-k}(\boldsymbol{\alpha}, \mathbf{v})$. Instead, we can use a basis of $C^{\perp}$, which is also a check matrix for C. Using a check matrix to define a linear code has its benefits. One such benefit is that it will allow us to use *syndrome decoding*, which will be discussed in more detail in a later section.

3 Encoding

4 Decoding

Having an encoded message is useless without someway to decode it. Reed-Solomon codes take advantage of a technique called *syndrome decoding*. Syndrome decoding uses the dual code and check matrices to decode and to detect and correct errors that may have occurred in transmission.

The code $\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$ over F is equal to the set of all n-tuples $\mathbf{c} = (c_1, c_2, ..., c_n) \in F^n$ such that:

$$\sum_{i=1}^{n} \frac{c_i u_i}{1-\alpha_i z} = 0 \pmod{z^r},$$

where $r = n - k$ and $u_i^{-1} = v_i \prod_{j \neq i} (\alpha_i - \alpha_j)$. This is known as the *Goppa formulation for GRS codes.* The Goppa formulation is simply another way of writing our code, but there are benefits to using this formulation as we will see below.

If $\mathbf{c} = (c_1, c_2, ..., c_n)$, a GRS code in the Goppa formulation, is transmitted, and $\mathbf{p} = (p_1, p_2, ..., p_n)$ is received, then there is some vector $\mathbf{e} = (e_1, e_2, ...e_n)$ such that $\mathbf{p} = \mathbf{c}+\mathbf{e}$. This vector is called the error vector. We can then calculate the *syndrome polynomial* of $\mathbf{p}$:

$$S_{\mathbf{p}}(z) = \sum_{i=1}^{n} \frac{p_i u_i}{1-\alpha_i^z} \pmod{z^r}.$$

We can do the same for $\mathbf{c}$ and $\mathbf{e}$ as well. The following equation holds true:

$$S_{\mathbf{p}}(z) = S_{\mathbf{c}}(z) + S_{\mathbf{e}}(z) \pmod{z^r}.$$

Since $\mathbf{c}$ is in the Goppa formulation

$$S_{\mathbf{c}}(z) = \sum_{i=1}^{n} \frac{c_i u_i}{1-\alpha_i z} = 0,$$

so we write:

$$S_{\mathbf{p}}(z) = S_{\mathbf{e}}(z) \pmod{z^r}.$$

Let B be the set of error locations:

$$B = \{i \mid e_i \neq 0\}.$$

Then the syndrome polynomial reduces further to:

$$S_{\mathbf{p}}(z) = S_{\mathbf{e}}(z) = \sum_{b \in B} \frac{e_b u_b}{1-\alpha_b z} \pmod{z^r}.$$

We can drop the subscripts and just write $S(z)$ for the syndrome polynomial.

The next step is to find the *Key Equation.* This is done by clearing the denominators:

$$\sigma(z)S(z) = \omega(z) \pmod{z^r},$$

where

$$\sigma(z) = \sigma_{\mathbf{e}}(z) = \prod_{b \in B} (1 - \alpha_b z)$$

and

$$\omega(z) = \omega\mathbf{e}(z) = \sum_{b \in B} e_b u_b \left(\prod_{a \in B, a \neq b} (1 - \alpha_a z) \right).$$

The polynomial $\sigma(z)$ is called the *error locator*, and the polynomial $\omega(z)$ is called the *error evaluator*.

We can determine the error vector $\mathbf{e}$ given $\sigma(z)$ and $\omega(z)$. If we assume for now that none of the a_i equal 0, then:

$$B = b \mid \sigma(a_b^{-1}) = 0$$

and, for each $b \in B$,

$$e_b = \frac{-\alpha_b \omega(\alpha_b^{-1})}{u_b \sigma'(\alpha_b^{-1})},$$

where $\sigma'(z)$ is the derivative of $\sigma(z)$. It can be shown that the polynomials $\sigma(z)$ and $\omega(z)$ determine the error vector even when some α_i is 0.

This method of decoding requires us to solve the Key Equation in order to find the error vector. The following theorem gives us a characterization of $\sigma(z)$ and $\omega(z)$, helping us solve the Key Equation.

Given r and $S(z) \in F[z]$ there is at most one pair of polynomials $\sigma(z), \omega(z) \in F[z]$ satisfying:

1. $\sigma(z)S(z) = \omega(z) \pmod{z^r}$;

2. $\deg(\sigma(z)) \leq r/2$ and $\deg(\omega(z)) < r/2$;

3. $\gcd(\sigma(z), \omega(z)) = 1$ and $\sigma(0) = 1$.

Using this characterization we can solve the Key Equation using the Euclidean algorithm. For a code $\mathrm{GRS}_{n,k}(\boldsymbol{\alpha}, \mathbf{v})$ over F, with $r = n-k$, given a syndrome polynomial $S(z)$, the following algorithm halts, producing $\tilde{\sigma}(z)$ and $\tilde{\omega}(z)$:

Algorithm 1: Decoding GRS using the Euclidean Algorithm

```
Set $a(z) = z^r$ and $b(z) = S(z)$.
Step through the Euclidean Algorithm until at Step $j$, $\deg(r_j(z)) < r/2$.
Set $\tilde{\sigma}(z) = t_j(z)$ and $\tilde{\omega}(z) = r_j(z)$.
```

If the error vector exists, then $\hat{\sigma}(z) = \tilde{\sigma}(0)^{-1}\tilde{\sigma}(z)$ and $\hat{\omega}(z) = \tilde{\sigma}(0)^{-1}\tilde{\omega}(z)$ are the error locator and error evaluator polynomials for $\mathbf{e}$. However, there are a few stipulations. The number of roots of $\hat{\sigma}(z)$ among the α_i^{-1} must be equal to the degree of $\hat{\sigma}(z)$. If this is not the case, that means we have detected errors which we cannot correct. Also, if $t_j(0) = 0$ we cannot perform the final division to determine $\hat{\sigma}(z)$.

If $t_j(0) \neq 0$ and $\hat{\sigma}(z)$ has the correct number of roots, then we can evaluate errors at each location to find a vector of weight at most $r/2$ with our original syndrome. At this point we have either decoded correctly, or we had more than $r/2$ errors and could not decode properly.