Encoding and Decoding Reed-Solomon Codes

John McCall

March 2, 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:

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

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