Error Correcting Codes
(Reed-Solomon encoding and Berlekamp-Welch decoding)

Grant McNaughton

13 April 2024

1 Introduction

Error correcting codes strengthen our ability to send and receive messages between two parties in the presence of errors caused by noise. They are typically used as follows: The sender encodes a message into a code and the receiver decodes the received code (with errors) by correcting the errors. Naturally, we want to ensure the following:

- *Correctness*: The message sent by the sender is correctly received by the receiver (almost always).
- *Efficiency*: Encoding should be efficient for the sender and decoding should be efficient for the receiver.

Usually, this is done by

- "encoding" the messages sufficiently different from each other
- "decoding" the encoded messages with errors by finding "nearby" message.

In this paper, we describe

1. encoding schemed from Reed and Solomon.
2. decoding scheme from Berlekamp and Welch.

2 Encoding (Reed-Solomon)

Reed-Solomon encoding was first introduced by Irving Reed and Gustave Solomon in 1960. It is commonly used in CDs, DVDs, Blu-Ray Discs, and QR codes, among many other things, to prevent data loss when transmitting by adding extra dimensionality that can be used to ensure correctness.

The Reed-Solomon encoding algorithm converts from input to output as follows:

- *Input*: Some message $m = (m_0, m_1, \ldots, m_{k-1}) \in F_q^k$
- *Output*: Some encoded message $c = (c_0, c_1, \ldots, c_{n-1}) \in F_q^n$

where $n > k$. It does this using the process described below.

Algorithm 1 (Reed-Solomon Encoding)**.**

1. *Fix* $a = (a_0, a_1, \ldots, a_{n-1}) \in F_q^n$ *where for all* $i \neq j$, $a_i \neq a_j$.
2. *Create a function* $f : \mathbb{R} \to \mathbb{R}$ *where* $f(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1}$.

3. *Set $c_i = f(a_i)$ for $i = 0, 1, \ldots, n-1$.*

Example 2.

Suppose we want to encode the message $m = (3, 5, 4)$ in F_{13} using $a = (1, 2, 3, 4, 5, 6)$. In this case, $q = 13, k = 3$, and $n = 6$.

1. $f(x) = m_0 + m_1 x + m_2 x^2 = 3 + 5x + 4x^2$
2. $c_0 = 3 + 5 + 4 = 12$
   $c_1 = 3 + 5 \cdot 2 + 4 \cdot 2^2 = 3 + 10 + 16 = 29 = 3$
   $c_2 = 3 + 5 \cdot 3 + 4 \cdot 3^2 = 3 + 15 + 36 = 54 = 2$
   $c_3 = 3 + 5 \cdot 4 + 4 \cdot 4^2 = 3 + 20 + 64 = 87 = 9$
   $c_4 = 3 + 5 \cdot 5 + 4 \cdot 5^2 = 3 + 25 + 100 = 128 = 11$
   $c_5 = 3 + 5 \cdot 6 + 4 \cdot 6^2 = 3 + 30 + 144 = 177 = 8$
3. $c = (12, 3, 2, 9, 11, 8)$

Theorem 3. *The minimum Hamming distance between any two encoded messages m and m', $d_{min} = d(\Phi(m), \Phi(m')) = n - (k-1)$ where Φ is the Reed-Solomon encoding and d is the Hamming distance.*

Proof. We will prove this first by showing that for any two messages $m \neq m'$, it holds that $d(\Phi(m), \Phi(m')) \geq n - (k-1)$. We will then prove that there exists some $m \neq m'$ such that $d(\Phi(m), \Phi(m')) = n - (k-1)$ to complete our proof.

1. Let $m \neq m' \in F_q^k$ and define $c = \Phi(m)$ and $c' = \Phi(m')$. In this case, $d(\Phi(m), \Phi(m'))$ is equal to the number of nonzero entries in $c - c'$.

   Establish $g(x) = f(x) - f'(x) = (m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1}) - (m'_0 + m'_1 x + m'_2 x^2 + \cdots + m'_{k-1} x^{k-1}) = (m_0 - m'_0) + (m_1 - m'_1)x + \cdots + (m_{k-1} - m'_{k-1})x^{k-1}$. Note that if for all i, $m_i \neq m'_i$, then g is of degree $k-1$. If there exists some i such that $m_i = m'_i$, then g is of degree less than $k-1$. Thus $\deg(g) \leq k-1$. By the Fundamental Theorem of Algebra, g has at most $k-1$ roots.

   $c - c' = (f(a_0) - f'(a_0), f(a_1) - f'(a_1), \ldots, f(a_{n-1}, f'(a_{n-1})) = ((g(a_o), g(a_1), \ldots, g(a_{n-1}))$. $c - c'$ has n entries and thus the number of nonzero entries in $c - c'$ is equal to n minus the number of zero entries in $((g(a_o), g(a_1), \ldots, g(a_{n-1}))$.

   Because for all i, j, $a_i \neq a_j$, there are at most $k-1$ zero entries in $((g(a_o), g(a_1), \ldots, g(a_{n-1}))$, as there are at most $k-1$ roots of g. We will denote this as $g_{\text{zero entries}}$. From this,

   $$\begin{aligned} g_{\text{zero entries}} &\leq k-1 \\ -g_{\text{zero entries}} &\geq -(k-1) \\ d(\Phi(m), \Phi(m')) = d(c, c') = n - g_{\text{zero entries}} &\geq n - (k-1) \end{aligned}$$

   Thus $d(\Phi(m), \Phi(m')) \geq n - (k-1)$.

2. We now want to find a combination of messages m and m' such that the number of zero entries in $c - c'$ is exactly $k-1$. Thus the g generated by m and m' will have exactly $k-1$ roots and we can write g in the form $g(x) = (x - a_0)(x - a_1) \ldots (x - a_{k-2})$.

   Let $m' = (0, 0, \ldots, 0)$. Then $f'(x) = 0$ and $c' = (0, 0, \ldots, 0)$. Because $g = f - f'$ and $f'(x) = 0$, we can say that $g(x) = f(x)$ and m is defined as the coefficients of $g = f$. Thus $m = (m_0, m_1, \ldots, m_{k-2}, 1) \neq (0, 0, \ldots, 0) = m'$. Thus there exists some $m \neq m'$ where $d(\Phi(m), \Phi(m')) = n - (k-1)$

Because $d_{\min} \geq n - (k-1)$ and there exists some $d = n - (k-1)$, we can say that $d_{\min} = n - (k-1)$. $\square$

Theorem 4. *If $n \geq 2\epsilon + k$ where ϵ is a rough maximum for the number of transmission errors, then d_{min} is sufficiently large.*

Proof. Assume that we establish $d_{\text{min}} > 2\epsilon$ where ϵ is a hyperparameter defining the amount of corruption expected during transmission. We have previously determined that $d_{\text{min}} = n - (k-1)$. From this, we can say that

$$\begin{aligned} d_{\text{min}} = n - (k-1) &> 2\epsilon \\ n &> 2\epsilon + k - 1 \end{aligned}$$

Thus any $n \geq 2\epsilon + k$ is "sufficiently large" enough for our d_{min}.

□

3 Decoding (Berlekamp-Welch)

The Berlekamp-Welch algorithm was developed by Elwyn Berlekamn and Lloyd Welch in 1986 as an efficient method to decode Reed-Solomon encoding. The Berlekamp-Welch algorithm converts from input to output as follows:

- *Input*: Some (possibly corrupted) encoded message $b \in F_q^n$
- *Output*: Some decoded message $m^* \in F_q^k$ such that $d(\Phi(m^*), b) \leq \epsilon$, if such an m^* exists

This algorithm will ideally yield $m^* = m$ or an error if the message is too corrupted, but there is a small possibility that a specific combination of corrupting errors may return $m^* \neq m$. The likelihood of this happening is very low, as the error would have to force m^* to look like a completely different valid message in $\mathbb{R}^n$ space.

This algorithm requires us to establish two polynomials E and G. E is our error-locating polynomial and must be such that $E(a_i) = 0$ if and only if $b_i \neq c_i = f(a_i)$ and that $E(x) = \Pi_{j \text{ s.t. } b_j \neq c_j}(x - a_i)x^{\epsilon - d(c,b)}$. We will then define $G(x) = f(x)E(x)$.

E and G thus satisfy the following:

1. E is monic
2. $\deg(E) = d(b,c) + (\epsilon - d(c,b)) = \epsilon$
3. $\deg(G) = \deg(f) + \deg(E) = k - 1 + \epsilon$
4. For $) \leq i \leq n-1$, $b_i E(a_i) = G(a_i)$. This is shown by the following cases:
   (a) $c_i \neq b_i$:
   $b_i E(a_i) = b_i \cdot 0 = 0$
   $G(a_i) = f(a_i)E(a_i) = f(a_i) \cdot 0 = 0$
   Thus $b_i E(a_i) = 0 = G(a_i)$.
   (b) $c_i = b_i$:
   $G(a_i) = f(a_i)E(a_i) = c_i E(a_i) = b_i E(a_i)$

Theorem 5. *If E, G and E', G' are two different sets of polynomials that both satisfy the properties above, then $\frac{E}{G} = \frac{E'}{G'}$.*

Proof. It suffices to show that $GE' - G'E = 0$, which can then be easily manipulated to show that $\frac{E}{G} = \frac{E'}{G'}$.

Let $R(a_i) = G(a_i)E'(a_i) - G'(a_i)E(a_i)$. Thus $\deg(R) \leq \max(\deg(GE'), \deg(G'E))$. As defined above, $\deg(E) = \deg(E') = \epsilon$ and $\deg(G) = \deg(G') = k - 1 + \epsilon$. From this, we can say that $\deg(GE') = \deg(G'E) = \deg(G) + \deg(E) = k - 1 + 2\epsilon$.

Because $\deg(R) = \max(\deg(GE'), \deg(G'E)) = \max(k - 1 + 2\epsilon, k - 1 + 2\epsilon) = k + 2\epsilon - 1 = n - 1$, we know that R can have at most $n - 1$ roots without being the trivial polynomial $R(a_i) = 0$. However, R must have n solutions, so $R = GE' - G'E = 0$. □

We will now describe the algorithm used for Berlekamp-Welch decoding from our polynomials E and G as well as our corrupted message b. Note the following:

- $E(x) = s_0 + s_1 x + s_2 x^2 + \cdots + s_{\epsilon-1}x^{\epsilon-1} + 1x^{\epsilon}$
- $G(x) = t_0 + t_1 x + t_2 x^2 + \cdots + t_{k-1+\epsilon}x^{k-1+\epsilon}$
- $b_i E(a_i) = G(a_i)$

We can then expand this into a system of equations:

$$b_0(s_0 + s_1 a_0 + \cdots + s_{\epsilon-1}a_0^{\epsilon-1} + a_0^{\epsilon}) = t_0 + t_1 a_0 + \cdots + t_{k-1+\epsilon}a_0^{k-1+\epsilon}$$
$$\vdots$$
$$b_{n-1}(s_0 + s_1 a_{n-1} + \cdots + s_{\epsilon-1}a_{n-1}^{\epsilon-1} + a_{n-1}^{\epsilon}) = t_0 + t_1 a_{n-1} + \cdots + t_{k-1+\epsilon}a_{n-1}^{k-1+\epsilon}$$

Rearranging to isolate our known term $b_n a_n^{\epsilon}$, we have

$$b_0 s_0 + b_0 s_1 a_0 + \cdots + b_0 s_{\epsilon-1}a_0^{\epsilon-1} - t_0 - t_1 a_0 - \cdots - t_{k-1+\epsilon}a_0^{k-1+\epsilon} = -b_0 a_0^{\epsilon}$$
$$\vdots$$
$$b_{n-1}s_0 + b_{n-1}s_1 a_{n-1} + \cdots + b_{n-1}s_{\epsilon-1}a_{n-1}^{\epsilon-1} - t_0 - t_1 a_{n-1} - \cdots - t_{k-1+\epsilon}a_{n-1}^{k-1+\epsilon} = -b_{n-1}a_{n-1}$$

We can then rewrite this system of n unknowns in n equations as a matrix problem $Uz = V$

$$\begin{bmatrix} b_0 a_0^0 & \cdots & b_0 a_0^{\epsilon-1} & -a_0^0 & \cdots & -a_0^{k-1+\epsilon} \\ \vdots & & & & & \\ b_{n-1}a_{n-1}^0 & \cdots & b_{n-1}a_{n-1}^{\epsilon-1} & -a_{n-1}^0 & \cdots & -a_{n-1}^{k-1+\epsilon} \end{bmatrix} \begin{bmatrix} s_0 \\ \vdots \\ s_{\epsilon-1} \\ t_0 \\ \vdots \\ t_{k-1+\epsilon} \end{bmatrix} = \begin{bmatrix} -b_0 a_-^{\epsilon} \\ \vdots \\ -b_{n-1}a_{n-1}^{\epsilon} \end{bmatrix}$$

where $U \in \mathbb{R}^{n\times n}$, $z \in \mathbb{R}^{n\times 1}$, and $V \in \mathbb{R}^{n\times 1}$. Thus we can find all our coefficients for E and G by solving the straightforward system of equations above. Lastly, we can work recursively from our definitions to finish the algorithm.

Algorithm 6. *Berlekamp-Welch decoding*

1. $E, G \leftarrow$ *Found as shown above*
2. $f^* \leftarrow \frac{G}{E}$
3. $m^* \leftarrow$ *Coefficients of* f^*
4. *If* $d(\Phi(m^*), b) > \epsilon$, *return "Fail"*
5. $m \leftarrow m^*$