

Lecture 27: Berlekamp-Welch Algorithm

October 31, 2007

Lecturer: Atri Rudra

Scribe: Michel Kulhandjian

In the last lecture, we discussed unique decoding of RS codes and briefly went through the Berlekamp-Welch algorithm. In today's lecture we will study Berlekamp-Welch algorithm in more detail.

0.1 Decoding of Reed-Solomon Codes

Recall that the $[n, k, n-k+1]_q$ Reed-Solomon code regards a message as a polynomial p of a degree at most $k-1$, and the encoding of a message $\mathbf{m} = \langle m_0, \dots, m_{k-1} \rangle$ is $\langle p(\alpha_1), \dots, p(\alpha_n) \rangle$. Here, $m_i \in \mathbb{F}_q$, $k \leq n \leq q$, and $p(x) = \sum_{i=0}^{k-1} m_i x^i$. Upon passing through a noisy channel we receive $\mathbf{y} = (y_1, \dots, y_n)$. Now let us look at the decoding problem of Reed-Solomon codes. Suppose we are given distinct values $\alpha_1, \dots, \alpha_n$ where $\alpha_i \in \mathbb{F}_q$ with received vector $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}_q^n$ and parameters k and $e < \frac{n-k+1}{2}$, where e is the number of errors which occurred in the received vector. Our goal is to find a polynomial $p \in \mathbb{F}_q[x]$ of degree $\leq k-1$, such that $p(\alpha_i) \neq y_i$ for at most e values of $i \in [n]$. Although this problem is quite non-trivial one, a polynomial time solution can be found for this problem. This solution dates back to 1960 when Peterson [2] came up with a decoding algorithm for the more general BCH code that runs in time $O(n^3)$. Later Berlekamp and Massey speeded up this algorithm so that it runs in $O(n^2)$. There is an implementation using Fast Fourier Transform that runs in time $O(n \log n)$. We will not give these faster algorithms, but actually study the Berlekamp-Welch algorithm (Gemmell - Sudan description [1]).

1 Berlekamp-Welch Algorithm

We start by describing the major steps in the Berlekamp-Welch algorithm.

Input: $n \geq k \geq 1$, $e < \frac{n-k+1}{2}$ and n pairs $\{(\alpha_i, y_i)\}_{i=1}^n$ with α_i distinct. Let $\mathbf{y} = (y_1, \dots, y_n)$

Output: Polynomial $p(x)$ or "fail".

Step 1: Compute a non-zero polynomial $N(x)$ of degree $= e$, and a polynomial $Q(x)$ of degree $\leq e + k - 1$ such that

$$y_i N(\alpha_i) = Q(\alpha_i) \quad 1 \leq i \leq n \quad (1)$$

if such polynomials do not exist output "fail".

Step 2: If $N(x)$ does not divide $Q(x)$ output "fail" else output $\frac{Q(x)}{N(x)}$.

Remark 1.1. *May not know e so we need to run the algorithm for $0 \leq e < \frac{n-k+1}{2}$.*

We first prove the correctness of the Berlekamp-Welch algorithm:

Theorem 1.2. *If $\langle p(\alpha_i) \rangle_i$ is transmitted and $\leq e$ error occur, then the Berlekamp-Welch algorithm outputs $p(x)$ with degree $\leq k - 1$.*

Claim 1.3. *There exist a pair of $N(x)$ and $Q(x)$ that satisfy Step 1 such that $\frac{Q(x)}{N(x)} = p(x)$.*

Note that it suffices to argue that $\frac{Q_1(x)}{N_1(x)} = \frac{Q_2(x)}{N_2(x)}$ for any pair of solutions that satisfy Step 1 since the Claim 1.3 above can then be used to see that ratio must be $p(x)$.

Claim 1.4. *If any two distinct solutions $(N_1(x), Q_1(x)) \neq (N_2(x), Q_2(x))$ satisfy Step 1, then they will satisfy*

$$\frac{Q_1(x)}{N_1(x)} = \frac{Q_2(x)}{N_2(x)}.$$

Proof of Claim 1.3 Note that as both $N_1(x) \neq 0$ and $N_2(x) \neq 0$, $Q_1(x)N_2(x) = Q_2(x)N_1(x)$. We just take $N(x)$ to be error-locating polynomial for $p(x)$ and let $Q(x) = p(x)N(x)$ where $\deg(Q(x)) \leq \deg(p(x)) + \deg(N(x)) \leq e + k - 1$. Here is a polynomial $N(x)$ that satisfies above and has degree of exactly e .

$$N(x) = x^{e-\Delta(\mathbf{y}, \langle p(\alpha_i) \rangle_i)} \prod_{1 \leq i \leq n | y_i \neq p(\alpha_i)} (x - \alpha_i) \quad (2)$$

By definition, $N(x)$ is a degree e polynomial with the following property:

$$N(\alpha_i) = 0 \quad \text{iff} \quad y_i \neq p(\alpha_i)$$

We now argue that $N(x)$ and $Q(x)$ satisfy (1). Note that if $N(\alpha_i) = 0$, then $p(\alpha_i)N(\alpha_i) = y_i N(\alpha_i) = 0$. When $N(\alpha_i) \neq 0$, we know $p(\alpha_i) = y_i$ and so we still have $p(\alpha_i)N(\alpha_i) = y_i N(\alpha_i)$ \square

Notice that $N(x)$ error-locating polynomial is as hard to find as the solution polynomial $p(x)$. If one knew $N(x)$, then one can use erasure decoding for RS codes to recover $p(x)$. Also note that polynomial $Q(x)$ is as hard to find as $N(x)$. Given $Q(x)$ and \mathbf{y} one can find the error locations by checking positions where $y_i \neq 0$ and $Q(\alpha_i) = 0$. While each of these polynomials $N(x)$, $Q(x)$ is hard to find individually, together they are easier to find.

Proof of Claim 1.4 Note that the degrees of the polynomials $Q_1(x)N_2(x)$ and $Q_2(x)N_1(x)$ are at most $e + k - 1$. Lets define polynomial $R(x)$ with degree $\leq e + k - 1 + e$ as follows,

$$R(x) = Q_1(x)N_2(x) - Q_2(x)N_1(x). \quad (3)$$

Furthermore, from Step 1 we have, for every $i \in [n]$,

$$y_i N_1(\alpha_i) = Q_1(\alpha_i) \quad \text{and} \quad y_i N_2(\alpha_i) = Q_2(\alpha_i). \quad (4)$$

Substituting (4) into (3) we get for $1 \leq i \leq n$:

$$\begin{aligned} R(\alpha_i) &= (y_i N_1(\alpha_i))N_2(\alpha_i) - (y_i N_2(\alpha_i))N_1(\alpha_i) \\ &= 0 \end{aligned}$$

The polynomial $R(x)$ has n roots and

$$\begin{aligned} \deg(R(x)) &\leq e + k - 1 + e \\ &= 2e + k - 1 \\ &< n \end{aligned}$$

Where the last inequality follows from the upper bound one. Since $\deg(R(x)) < n$, then we get that the polynomials $Q_1(x)N_2(x)$ and $Q_2(x)N_1(x)$ agree on more points than their degree, and hence they are identical, as desired. Claim 1.3 and 1.4 prove Theorem 1.2. \square

1.1 Implementation of Berlekamp-Welch Algorithm

In *Step 1*, $Q(x)$ has $e + k$ unknowns and $N(x)$ has $e + 1$ unknowns. For each $1 \leq i \leq n$ constrains by equation (1) is a linear equation in these unknowns (homogeneous).

System of n linear equations in $2e + k + 1$ unknowns $< n + 2$. By claim 1.3, these system of equations have a solution, and can be solved in $O(n^3)$ time. The only extra requirement is that the degree of polynomial $N(x)$ should be exactly e . We have already shown $N(x)$ in equation (2) that satisfy this requirement. So we add a constraint that the coefficient of x^e in $N(x)$ is 1. We have $n + 1$ linear equation in $\leq n + 1$ variables. Finally, note that *Step 2* can be implemented in time $O(n^3)$ by “long division”.

Theorem 1.5. *For any $[n, k]_q$ Reed-Solomon code it can be uniquely decoded in $O(n^3)$ time for $< \frac{D}{2} = \frac{n-k+1}{2}$ number of errors.*

2 Errors And Erasures decoding of RS codes

There exists polynomial time constructible codes that lie on the Zyablov bound and can be corrected upto $1/4$ of their designed distance(similar results for Justesen codes). Next, we look into codes that correct up to $\frac{1}{2}$ design distance for explicit codes on the Zyablov bound. For this one we need correction from both *errors* and *erasures*. Berlekamp-Welch algorithm corrects upto $e < \frac{D}{2}$ errors for $RS [N, K, D]_Q$. We have already seen that we can correct $S < D$ erasures. The number of unknown position is S and the number of known position is $N - S \geq N - D + 1 = K$. Having K constrains we can recover polynomial $\deg \leq K - 1$ by interpolation (K unknowns and K equations) in $O(N^3)$ time.

Proposition 2.1. *Reed-Solomon codes can be corrected from e errors and S erasures as long as $2e + S < D$ in $O(N^3)$ time.*

This result will be proved in the next lecture.

References

- [1] V. Guruswami and M. Sudan. Improved decoding of reed-solomon codes and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, Sept. 1999.
- [2] W. Wesley Peterson. Encoding and error-correction procedures for bose-chadhuri codes. *IEEE Transactions on Information Theory*, 6:459–470, 1960.