COMPSCI 690T Coding Theory and Applications Mar 8, 2017

Lecture 12

Instructor: Arya Mazumdar *Scribe: Names Redacted*

1 More on Berlekamp-Welch Decoding

1.1 Reed-Solomon Codes

Last time we introduced a family of linear q-ary codes called Reed-Solomon codes. Recall that such a code is defined by first fixing a defining set $\{\alpha_1, \ldots, \alpha_n\}$ of distinct elements in $\mathbb{F}_q$ (which requires $q > n$), and then we define the code $\mathcal{C} = \{eval(f) : \deg(f) < k\}$, where $eval(f)$ is the vector $\big(f(\alpha_1) \cdots f(\alpha_n)\big)$.

We saw that these codes have $d = n - k + 1$, so are MDS (meet the Singleton bound with equality).

1.2 Berlekamp-Welch Decoding

We also saw a way to efficiently decode these codes. If we receive $\mathbf{r} = \big(r_1 \cdots r_n\big)$, we proved that we can always find a polynomial $Q(x, y) \in \mathbb{F}_q(x, y)$ such that

1. $Q(x, y) = Q_0(x) + yQ_1(x)$.
2. $\deg(Q_0) \leq n - t - 1$, and $\deg(Q_1) \leq n - t - 1 - (k - 1)$.
3. $Q(\alpha_i, r_i) = 0$ for $1 \leq i \leq n$.

Then to decode, we let $f(x) = -\frac{Q_0(x)}{Q_1(x)}$, and decode to $eval(f)$. We also showed last time that if there are at most $\frac{n-k}{2}$ errors, this decoding will be correct.

1.3 Intuition for Berlekamp-Welch Decoding

In general when we try to decode Reed-Solomon codes, each codeword is an evaluation of a polynomial of degree at most $k - 1$, on n different points. If there are $t \leq \frac{n-k}{2}$ errors, then the received word still must match some polynomial of degree at most $k - 1$ in at least $n - t$ points, so our goal is to find a curve in F_q that matches the received word in at least $n - t$ points.

This is an interpolation problem, and we have more equations than unknowns, so the difficulty is only in doing this quickly. If we don't care about time, we can just take every set of $n - t$ points, and try and fit a curve to it. We will only be able to do this successfully if the subset we pick contains all the errors, so when we find a curve that works, we will know where all t errors are. Unfortunately there are $\binom{n}{t}$ such subsets, so this is too slow.

By working backwards, we can get some intuition for why evaluating $-\frac{Q_0(x)}{Q_1(x)}$ is the right thing to do. Assume $f(x)$ is the correct polynomial, and we receive $\mathbf{r} = \big(r_1 \cdots r_n\big)$. First, we define the error locator polynomial to be

$$Q_1(x) = \prod_{\substack{1 \leq i \leq n \\ f(\alpha_i) \neq r_i}} (x - \alpha_i),$$

so that $Q_1(\alpha_i) = 0$ whenever $f(\alpha_i) \neq r_i$, i. e., whenever there is an error at coordinate i.

Then we have that for any i, $Q_1(\alpha_i)f(\alpha_i) = Q_1(\alpha_i)r_i$, because if $f(\alpha_i) = r_i$ clearly both sides are equal, and if not, then $Q_1(\alpha_i) = 0$, so both sides are 0. Then we have $r_i = -\frac{Q_0(\alpha_i)}{Q_1(\alpha_i)}$ whenever $Q_1(\alpha_i) \neq 0$, so we are looking for a solution to the equation $y = -\frac{Q_0(x)}{Q_1(x)}$.

2 List Decoding

2.1 Berlekamp-Welch Generalization (Sudan 1996)

List decoding is a generalization of standard decoding, where rather than returning a single codeword which we guarantee is the correct codeword, we return a list of $\leq L$ codewords and guarantee that one of them is the correct codeword. In many cases list decoding will allow us to decode in the presence of a greater number of errors than standard decoding.

In the previous lecture notes, it is shown how to generalize Berlekamp-Welch decoding for L-list decoding. Given a defining set $\{\alpha_1, \ldots, \alpha_n\}$ and received word $\mathbf{r} = (r_1 \cdots r_n)$, we find a polynomial $Q(x, y)$ such that

1. $Q(x, y) = Q_0(x) + yQ_1(x) + y^2Q_2(x) + \cdots + y^LQ_L(x)$.
2. $\deg(Q_j) \leq n - t - 1 - j(k-1)$ for $0 \leq j \leq L$.
3. $Q(\alpha_i, r_i) = 0$ for $1 \leq i \leq n$.

Then, we find all $f(x)$ that are y-roots of $Q(x, y)$, and return as our list $eval(f)$ for all such f. It is shown in the previous lecture notes that we can find such a polynomial and that the correct codeword $f(x)$ will be a y-root, provided that the number of bits in error is $t < \min(\frac{nL}{L+1} - \frac{(k-1)L}{2}, n - L(k-1))$.

2.2 2-List Decoding Example

As an example, we can compute when list decoding with $L = 2$ will allow us to decode in the presence of more errors than regular decoding. This will occur when

$$\frac{n-k}{2} < \min(\frac{2n}{3} - (k-1), n - 2(k-1)).$$

Doing some algebra, we see that $\frac{n-k}{2} < \frac{2n}{3} - (k-1)$ implies $\frac{k}{n} < \frac{1}{3} - \frac{1}{2n}$, and $\frac{n-k}{2} < n - 2(k-1)$ implies $\frac{k}{n} < \frac{1}{3} + \frac{1}{3n}$. For large n both of these values go to $\frac{1}{3}$, so we can decode from more errors with 2-list decoding as long as the rate of our code is less than $\frac{1}{3}$. Figure 1 shows the two constraints on t for 2-list decoding, and the shaded region represents achievable pairs of values for $\frac{t}{n}$ and $\frac{k}{n}$.

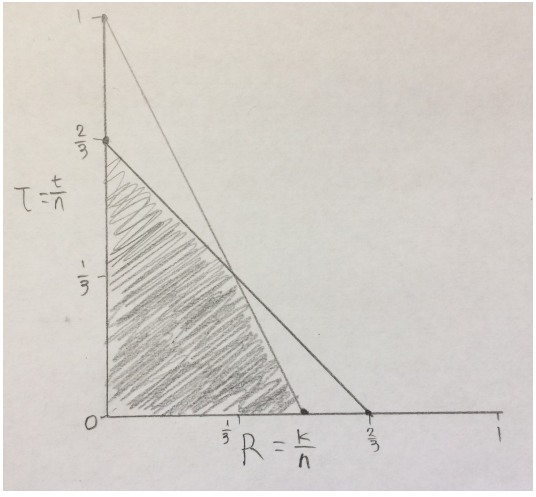

Figure 1: The attainable region for 2-list decoding.

3 Concatenated Codes

Recall that in a Reed-Solomon Code, the cardinality of the defining set must be less than the size of the field, i.e., $q > n$ for defining set $\{\alpha_1, \ldots, \alpha_n\}$ over $\mathbb{F}_q$. In practice, it is inconvenient for the field size to grow with the size of the defining set. For example, our analysis has been with respect to transmitting symbols, but in practice these would need to be encoded to binary, but then a single bit error would cause a symbol error, losing $\log_2(q)$ information bits.

Concatenated codes are used to overcome this inconvenience, by re-encoding the q-ary Reed-Solomon code to a binary code.[1] More formally, we define a $[n_o, k_o, d_o]_q = 2^j$ Reed-Solomon code $\mathcal{C}_o$, which we call the "outer code", and a $[n_i, k_i, d_i]_2$ linear "inner code" $\mathcal{C}_i$, such that there is a one-to-one mapping between the q-ary alphabet of the outer code and the binary codewords of the inner code. This injection implies that the dimension of the inner code, k_i, is equal to $\log_2 q$ and the rate of the inner code, R_i, is equal to $\frac{\log_2 q}{n_i}$.

We encode by taking the outer codeword symbols and multiplying them by the generator matrix of the inner code. The resultant concatenated code is a linear code of length $n_o n_i$. We can compute the dimension by taking $\log_2$ of the number of codewords of the outer code, and see that $\log_2(|\mathcal{C}_o|) = \log_2(q^{k_o}) = k_o \log_2(q) = k_o k_i$. The minimum weight nonzero codeword must have at least d_o nonzero outer code symbols, and each of these symbols will have at least d_i nonzero bits, as they are inner codewords, so the minimum distance of the code is $d_o d_i$.

[1] More generally, concatenated codes are also capable of re-encoding the q-ary code to b-ary, though it is most efficient when $q = b^j$ for some j.