Introduction to Coding Theory **CMU: Spring 2010**

Notes 11: List Decoding Folded Reed-Solomon Codes

April 2010

Lecturer: Venkatesan Guruswami *Scribe: Venkatesan Guruswami*

At the end of the previous notes, we defined folded Reed-Solomon codes, parameterized by an integer folding parameter $s \geq 1$, as follows.

Definition 1 (s-folded RS code) *Let $\mathbb{F}$ be a field of size q whose nonzero elements are $\{1, \gamma, \ldots, \gamma^{n-1}\}$ for $n = q - 1$. Let $s \geq 1$ be an integer which divides n. Let $1 \leq k < n$ be the degree parameter.*

The folded Reed-Solomon code $\mathrm{FRS}^{(s)}_{\mathbb{F}}[k]$ is a code over alphabet $\mathbb{F}^s$ that encodes a polynomial $f \in \mathbb{F}[X]$ of degree k as

$$f(X) \mapsto \left(\begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{s-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^{s}) \\ f(\gamma^{s+1}) \\ \vdots \\ f(\gamma^{2s-1}) \end{bmatrix}, \ldots, \begin{bmatrix} f(\gamma^{n-s}) \\ f(\gamma^{n-s+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix}, \right) . \tag{1}$$

Observe that the folded RS code is a code of block length $N = n/s$ and rate $R = k/n$ (which is the same as the original, unfolded Reed-Solomon code).

1 List decoding FRS codes

Now suppose such a codeword was transmitted and we received a string in $\mathbf{y} \in (\mathbb{F}^s)^N$ which we view in matrix form as

$$\begin{array}{ccc} y_1 & y_{s+1} & y_{n-s+2} \\ y_2 & y_{s+2} & \vdots \\ y_3 & y_{s+3} & \vdots \\ & & \ddots \\ y_s & \cdots & y_n \end{array} \tag{2}$$

We would like to recover a list of all polynomials $f \in \mathbb{F}[X]$ of degree k whose folded RS encoding (1) agrees with $\mathbf{y}$ in at least t columns, for some agreement parameter t. The goal would be to give an algorithm for as small a t as possible, as this corresponds to list decoding up to $n - t$ errors. Note that we can solve this problem for $t \geq \sqrt{R}N$ by simply unfolding the received word and using the algorithm for Reed-Solomon codes. Our hope is to do better by exploiting the fact that the agreements have some structure: when a column is correct we know the correct values of *all* s values in the column. In other words, the number of errors patterns we have to worry about is smaller and they have some structure, which could be potentially be exploited to correct from a larger number of errors. This is in fact what we will do, exploiting in a crucial way the algebraic structure of the folding.

As in the Reed-Solomon list decoding algorithm, we will use interpolation to fit a low-degree nonzero polynomial Q through the data. The gain will come by interpolating in more than two dimensions. This enables working with a smaller degree and offers the hope of leading to an algorithm that works with smaller agreement. The fact that an entire column is correct when there is no error will be used to argue that the interpolated polynomial Q and the message polynomial f must obey some identity (the higher-dimensional analog of the identity $Q(X, f(X)) = 0$ from the Reed-Solomon case). The algebraic crux is to argue that this identity suffices to pin down f to a small list of possibilities, and to find this list efficiently.

1.1 Interpolation step

We will describe a decoding algorithm that can be viewed as a higher-dimensional generalization of the Welch-Berlekamp algorithm. The algorithm will interpolate a "linear" polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$ which has degree 1 in the Y_i's through certain $(s+1)$-tuples. We thank Salil Vadhan for pointing out this variant of the algorithm, which is simpler to describe. The algorithm described in the paper [1] uses higher degrees in Y_i's as well as multiplicities in the interpolation. This leads to a stronger bound (as we will mention later), but the simpler "linear polynomial" based algorithm suffices for the main message of this lecture. Jumping ahead, this message is the explicit construction of codes of rate R which can be list decoded in polynomial time from a fraction $1 - R - \epsilon$ of errors, for any desired $\epsilon > 0$.

Given a received word as in (2) we will interpolate a nonzero polynomial

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \cdots + A_s(X)Y_s$$

with the degree restrictions $\deg(A_i) \le D - 1$ for $i = 1, 2, \ldots, s$ and $\deg(A_0) \le D + k - 1$ for a suitable degree parameter. The number of monomials in a polynomial Q with these degree restrictions equals $Ds + D + k$. The polynomial $Q \in \mathbb{F}[X, Y_1, \ldots, Y_s]$ must satisfy the interpolation conditions

$$Q(\gamma^{is}, y_{is+1}, y_{is+2}, \cdots, y_{(i+1)s}) = 0 \quad \text{for } i = 0, 1, \ldots, n/s - 1 \ . \tag{3}$$

Provided $Ds + D + k > \frac{n}{s} = N$, or in other words

$$D > \frac{N-k}{s+1} \ , \tag{4}$$

such a nonzero polynomial Q must exist, and can be found in polynomial time by solving a homogeneous linear system. The following lemma shows that any such polynomial Q gives an algebraic condition that the message polynomials $f(X)$ we are interested in list decoding must satisfy.

Lemma 2 *If $f \in \mathbb{F}[X]$ is a polynomial of degree k whose FRS encoding (1) agrees with $\mathbf{y}$ in at least t columns for $t \ge D + k$, then*

$$Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) = 0 \ . \tag{5}$$

Proof: Define $R(X) = Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X))$. Due to the degree restrictions on Q, the degree of $R(X)$ is easily seen to be at most $D + k - 1$. If the FRS encoding of f agrees with $\mathbf{y}$ in the i'th column (for some $i \in \{0, 1, \ldots, N-1\}$), we have

$$f(\gamma^{is}) = y_{is+1}, \quad f(\gamma^{is+1}) = y_{is+2}, \quad \cdots \quad , f(\gamma^{is+s-1}) = y_{(i+1)s} \ .$$

Together with the interpolation conditions (3), this implies

$$R(\gamma^{is}) = Q(\gamma^{is}, f(\gamma^{is}), f(\gamma^{is+1}), \cdots, f(\gamma^{is+s-1})) = 0 \ .$$

Hence R has at least t zeroes. Since $\deg(R) \le D + k - 1$, if $t \ge D + k$, we must have $R = 0$. $\square$

1.2 Root-finding step

The question that arises now is how restrictive is Equation (5) in terms of pinning down the number of possible solutions $f(X)$ to a small (polynomial in $|\mathbb{F}|$) number? For purposes of illustration and ease of notation, let us focus on the $s = 2$ case, so that we need to find the list of all degree k solutions $f(X)$ to

$$Q(X, f(X), f(\gamma X)) = 0 \ . \tag{6}$$

For this part we will have use the fact that $\gamma \in \mathbb{F}$ is not arbitrary but is a primitive element. Indeed if $\gamma = 1$, then for the polynomial $Q(X, Y, Z) = Y - Z$, *every* polynomial f will satisfy $Q(X, f(X), f(X)) = 0$. Likewise, if $\gamma = -1$, then every polynomial $f(X) = g(X^2)$ will satisfy $Q(X, f(X), f(-X)) = 0$ for $Q(X, Y, Z) = Y - Z$. This argument shows that if the order of γ is small, then there can be too many ($|\mathbb{F}|^{\Omega(k)}$) degree k solutions.

We will next prove that for primitive γ, the number of f of degree less than $q - 1$ (and hence also number with degree at most k) satisfying (6) is at most q. For this we need two lemmas.

Lemma 3 *If* $\deg(f) < q - 1$, *then* $f(\gamma X) = f(X)^q \mod (X^{q-1} - \gamma)$.

Proof: We have $X^q \equiv \gamma X \pmod{X^{q-1} - \gamma}$, and therefore

$$f(X^q) \equiv f(\gamma X) \pmod{X^{q-1} - \gamma} \ .$$

For $f \in \mathbb{F}[X]$, we have $f(X^q) = f(X)^q$. Therefore $f(X^q) \equiv f(\gamma X) \pmod{X^{q-1} - \gamma}$. Since the degree of f is less than $q - 1$, the remainder of $f(X)^q \bmod X^{q-1} - \gamma$ must equal $f(\gamma X)$. $\square$

The next lemma (in fact a more general statement) was on problem set 2.

Lemma 4 *The polynomial* $X^{q-1} - \gamma$ *is irreducible over* $\mathbb{F}_q$ *when* γ *is a primitive element of* $\mathbb{F}_q$.

Denote $E(X) = X^{q-1} - \gamma$. Denote by L the extension field $L = \mathbb{F}[X]/(E(X))$ (it is a field since $E(X)$ is irreducible), and for a polynomial $f \in \mathbb{F}[X]$, let $\bar{f}$ denote its residue (modulo $E(X)$) in L.

Theorem 5 *Given a nonzero* $Q \in \mathbb{F}[X, Y_1, Y_2]$ *which is linear in* Y_1, Y_2, *the number of polynomials* $f \in \mathbb{F}[X]$ *of degree less than* $q - 1$ *such that* $Q(X, f(X), f(\gamma X)) = 0$ *is at most* q. *Moreover the list of all such polynomials can be found in time polynomial in* q *and the degree of* Q.

Proof: Factor out the largest power of $E(X)$ that divides Q, so that $Q(X, Y_1, Y_2) = E(X)^a Q_1(X, Y_1, Y_2)$ for some $a \ge 0$ and polynomial Q_1 that is not divisible by $E(X)$. Define the polynomial $S \in L[Y]$ as

$$S(Y) = Q_1(X, Y, Y^q) \mod E(X) \ .$$

Since Q_1 has degree in Y_1, Y_2 less than q (in fact it is linear in Y_1, Y_2), and it is not divisible by $E(X)$, we have $S \neq 0$. Clearly if $Q(X, f(X), f(\gamma X)) = 0$, then $Q_1(X, f(X), f(\gamma X)) = 0$, so certainly $Q_1(X, f(X), f(\gamma X)) \mod E(X) = 0$. By Lemma **??**, this implies $Q_1(X, f(X), f(X)^q) \mod E(X) = 0$, i.e., $S(\bar{f}) = 0$. Thus $\bar{f}$ must be a root of S, which has degree at most q in Y. This implies that there are most q solutions for $\bar{f}$. Since f has degree less than $q-1$, f can be uniquely recovered from $\bar{f}$, and hence there are at most q solutions f to $Q(X, f(X), f(\gamma X)) = 0$, as desired.

The claim about the running time follows since the polynomial S can be computed efficiently from Q, and there are efficient root finding algorithms known over large extension fields. (One can either have a randomized algorithm running in time polynomial in the degree and log of the field size, or a deterministic algorithm running in time polynomial in the degree, log of the field size, *and* the characteristic of the field. In our case, even the deterministic runtime will be polynomial in q.) $\square$

The above approach generalizes readily to the larger variate case. Any solution $f \in \mathbb{F}[X]$ to $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X)) = 0$ will have its residue $\bar{f}$ as a root of (the nonzero polynomial) $S(Y) = Q(X, Y, Y^q, \cdots, Y^{q^{s-1}}) \mod E(X)$. Therefore there will be at most q^{s-1} such polynomials f and they can all be found in polynomial (in q) time.

1.3 Decoding guarantee

Combining Lemma 2 and Theorem 5, and recalling the requirement (4) on the degree parameter D we conclude that we have a polynomial time list decoding algorithm for $\mathrm{FRS}_{\mathbb{F}}^{(s)}[k]$ to find all message polynomials whose encoding has agreement t with an input $\mathbf{y} \in (\mathbb{F}^s)^N$ provided

$$t > \frac{N-k}{s+1} + k = \frac{N}{s+1} + \frac{s}{s+1} k \ . \tag{7}$$

The fractional agreement τ required (in terms of the rate) is

$$\tau = \frac{t}{N} > \frac{1}{s+1} + \frac{k}{N} \frac{s}{(s+1)} = \frac{1}{s+1} + (sR) \frac{s}{s+1} \tag{8}$$

since $N = n/s$ and $R = k/n$.

Our goal was to decode from an agreement fraction $\tau \approx R$, but the factor s multiplying R implies that we require $\tau \approx sR$ which is much worse for large s. In particular we do not get anything meaningful for $R > 1/s$!

Remark 6 *By allowing higher degree terms in the Y_i's in the interpolated polynomial and using multiplicities in the interpolation (as in the improved Reed-Solomon list-decoding algorithm, extended to the multivariate case in the obvious manner), one can improve the above guarantee and decode from agreement fraction*

$$\tau \approx (sR)^{s/(s+1)} \ . \tag{9}$$

Note that this quantity is the geometric mean of $1, \underbrace{R, R, \cdots, R}_{s \text{ times}}$, *whereas the agreement required in (8) was the arithmetic mean of these values (which is in general larger). Recall that for Reed-Solomon codes ($s = 1$) this was also exactly the case: we reduced the agreement required from $\frac{1+R}{2}$ to $\sqrt{R}$ to get an algorithm to list decode up to the Johnson radius.*

Remark 7 (Parvaresh-Vardy) *The decoding guarantee (9) was obtained by Parvaresh and Vardy [2]. Their construction was somewhat different. Instead of the folding operation, they encoded a degree k message polynomial $f \in \mathbb{F}[X]$ by the evaluation of f and $(s-1)$ other polynomials $f_1, f_2, \ldots, f_{s-1}$ which are chosen to be algebraically related to $f(X)$ in the following way:*

$$f_i(X) = f(X)^{h^i} \mod E(X)$$

for $i = 1, 2, \ldots, s-1$, where $E(X)$ is an arbitrary irreducible polynomial of degree $k+1$. Note that the rate of such a code is R_0/s where R_0 is the rate of the original RS code (in particular this construction can never have rate exceed $1/s$). We used $f(\gamma^{i-1}X)$ in place of $f_i(X)$ in the above description in preparation for the optimal result which we discuss next.

2 Improving the decoding radius and correcting $1 - R - \epsilon$ errors

The idea to improve the agreement fraction required is to use folded codes with folding parameter m for $m \gg s$, but still only do $s+1$-variate interpolation. The key gain will be that each agreement location will now lead to many zeroes for the polynomial $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X))$ which ultimately ensures that this polynomial must equal zero even for a much smaller value of the agreement parameter t.

Specifically, let us consider the code $\mathrm{FRS}_{\mathbb{F}}^{(m)}[k]$, where as before $|\mathbb{F}| = q$, $n = q-1$, $m|n$, and the block length of the code equals $N = nm$. We will still interpolate a polynomial $Q \in \mathbb{F}[X, Y_1, \ldots, Y_s]$ in $s+1$ variables, but now instead of n/s tuples, we will interpolate through $(n/m)(m-s+1) = N(m-s+1)$ tuples given by

$$(\gamma^{im+j}, y_{im+j+1}, \ldots, y_{im+j+s}) \quad 0 \le i < n/m, \quad 0 \le j < m-s \ .$$

That is, for each column of m symbols, we interpolate through the $(m-s+1)$ windows of s consecutive symbols. The rationale behind this is that if we have an agreement in location i, i.e.,

$$(y_{im+1}, y_{im+2}, \ldots, y_{(i+1)m}) = (f(\gamma^{is}), f(\gamma^{is+1}), \ldots, f(\gamma^{im+m-1}))$$

then we get $m-s+1$ zeroes

$$Q(\gamma^{im+j}, f(\gamma^{im+j}), \ldots, f(\gamma^{im+j+s-1})) = 0 \quad j \in \{0, 1, 2, \ldots, m-s\} \ .$$

Thus t agreements leads to $t(m-s+1)$ zeroes for $R(X) = Q(X, f(X), \ldots, f(\gamma^{s-1}X))$. In place of (7), we get the new condition for successful decoding

$$t(m-s+1) > \frac{N(m-s+1)}{s+1} + \frac{s}{s+1}k$$

or equivalently

$$\tau = \frac{t}{N} > \frac{1}{s+1} + \frac{s}{s+1}\frac{m}{m-s+1}R \tag{10}$$

(recalling that $k = Rn = RNm$).

We now have our desired optimal trade-off between τ and R: picking $s > 1/\epsilon$ and $m \geq s^2$, the above condition (10) for successful decoding is met if

$$\tau \geq R + \epsilon \ .$$

Let us look at the parameters of the code. The alphabet size in $q^m \approx n^m = (Nm)^m$ which is $(N/\epsilon^2)^{O(1/\epsilon^2)}$ for the choice $m \approx 1/\epsilon^2$. The bound on list size we obtain is at most $q^{s-1} \approx (Nm)^{O(s)} = (N/\epsilon^2)^{O(1/\epsilon)}$ for the choice $s \approx 1/\epsilon$. The running time of the decoding algorithm is also dominated by the root-finding step, which takes $q^{O(s)}$ time as it has to find roots of a polynomial of degree at most q^s over an extension field of size at most q^q.

We can thus state our main theorem as follows.

Theorem 8 (Explicit codes achieving list decoding capacity) *For every $\epsilon > 0$ and $0 < R < 1$, there is a family of folded Reed-Solomon codes which have rate at least R and which can be list decoded up to a fraction $1 - R - \epsilon$ of errors in time $(N/\epsilon^2)^{O(\epsilon^{-1})}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\epsilon^2)^{O(1/\epsilon^2)}$.*

Remark 9 (Improvement using higher degree and multiplicities in interpolation) *Similar to Remark 6, by allowing higher degree terms in the Y_i's in the interpolated polynomial and using multiplicities in the interpolation, one can improve the above guarantee and decode from agreement fraction*

$$\tau \approx \left(\frac{mR}{m - s + 1} \right)^{s/(s+1)} \ . \tag{11}$$

instead of the arithmetic mean $\frac{1}{s+1} + \frac{s}{s+1} \frac{mR}{m-s+1}$. For details see the original paper [1].

References

[1] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008. 2, 6

[2] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005. 5