CS 229r Essential Coding Theory March 9, 2020

Lecture 12

Instructor: Madhu Sudan *Scribe: Jenny Kaufmann*

1 Folded Reed Solomon Codes

1.1 Definition

Folded Reed Solomon codes are a variant of Reed Solomon codes, defined as follows:

- Choose distinct field elements $\alpha_1, \dots \alpha_n \in \mathbb{F}_q$.
- Choose $\lambda \in \mathbb{F}_q^*$ of order $> k$.
- Choose a constant parameter r such that the $\lambda, \lambda^2, \dots \lambda^{r-1}$ are distinct.

Messages are polynomials $P(x)$ of degree $< k$ in the variable x, i.e. elements of $\mathbb{F}_q^{<k}[x]$. They are encoded as $[P(\alpha_i), P(\lambda\alpha_i), P(\lambda^2\alpha_i), \dots P(\lambda^{r-1}\alpha_i)]_{i=1}^n$. These codewords consist of n blocks of size r; they are elements of $(\mathbb{F}_q^r)^n$.

In particular, in folded Reed-Solomon codes, we count errors differently: whereas in an ordinary RS code, we count each incorrect character as one error, here we count each block containing an incorrect character as one error. Intuitively, we are thinking of the blocks as symbols in $\mathbb{F}_q^r$, instead of individual symbols.

Exercise 1. *What is the rate of this code?*

1.2 Decoding Idea

This code can be list decoded from $1 - R - \varepsilon$ fraction errors where $\varepsilon = \varepsilon(r) \to 0$ as $r \to \infty$. We will allow lists of size q^r.

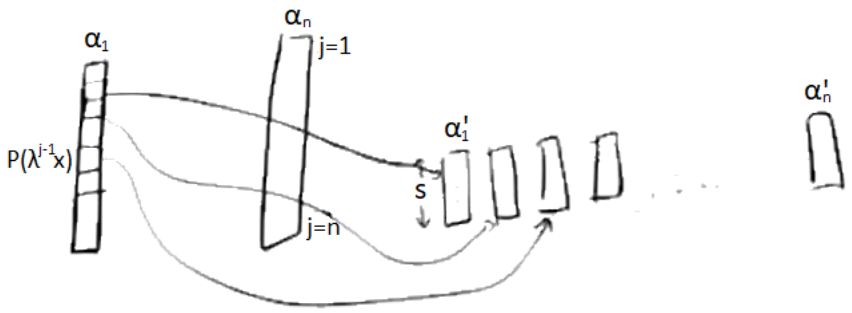
The idea behind the decoding is as follows: We are given $\alpha_1, \dots \alpha_n$. For each $i \in [n]$, we will create a column, with entries $\beta_i^1, \dots \beta_i^r$. Here, the upper index j represents the row. To obtain these β_i^j, we evaluate the polynomial $P(\lambda^{j-1} \cdot x)$ at the point $x = \alpha_i$.

This idea is based on earlier ideas by Coppersmith and Sudan, who originally devised an algorithm for list-decoding interleaved codes. The algorithm was improved by Parvaresh and Vardy, who suggested using polynomials related to each other in some way, and then further improved by Guruswami and Rudra, who suggested the specific construction we use here. Their reasoning was quite specialized and required that λ be a primitive element; later other work simplied the algorithms and used cleaner assumptions. In class we did not hear about this in much detail, but there is more information on the history of this algorithm here.

There is a much better analysis of the same codes - recent analysis: $O_\varepsilon(1)$, running time $O_\varepsilon(n^2)$.

Claim 2. *The algorithm below will find P if $m \triangleq \#\{i \mid \forall j \ P(\lambda^{j-1}\alpha_i = \beta_i^j\} \geq \frac{n}{r+1} + \frac{rk}{r+1}$, provided that λ has high order and are $\alpha_1, \dots \alpha_n$ distinct.*

For each column i corresponding to an element α_i, we generate many sub-column blocks via what one might refer to as "superadditive splintering." Each subcolumn corresponds to some value $\alpha_i' = \alpha_i\lambda^{j-1}$ for some values of j; the values of j may be overlapping between different blocks (hence "superadditive"), and hence some of the evaluation points in one block are the same as some in others.



For each column, we generate $r - s$ blocks, for some choice of $s << r$ (say $s^2 < r$). Then $n' = n(r - s)$ is the number of blocks in total.

This conversion from n columns to n' columns has not changed k: the message space has not changed.

Suppose a column is the original set of columns contains an error, that is, a point at which $P(\lambda^{j-1}\alpha_i \neq \beta_i^j$. This column splinters into $r - s$ columns in the new version, so if it contains an error, this can produce at most $r-s$ errors in the new version. Thus, if there is a certain fraction ρ of errors (that is, columns containing an error) in the original version, there is at most the same fraction ρ of errors in the new version, since only blocks deriving from columns with errors in the original can have errors in the new version. Equivalently, if the number of correct evaluation points in the original version is m, the number of correct evaluation points is $m(r - s)$ in the new version.

It is possible to find P (i.e., correct errors) if the number $m(r-s)$ of correct evaluation points is $\geq \frac{n(r-s)}{s+1} + \frac{sk}{s+1}$. Then, if $r > s^2$,

$$m \geq \frac{n}{s+1} + \frac{sk}{(r-s)(s+1)} \leq \frac{2n}{s} + \frac{k}{r}.$$

1.3 Decoding Algorithm

The algorithm is as follows: Say we are given a parameter $D = \frac{n}{r+1} - \frac{k}{r+1}$. Then:

1. Find $A_0, \ldots A_r \in \mathbb{F}_q[x]$ not all zero such that $\forall i$, $A_0(\alpha_i) + \sum_{j=1}^{r} \beta_i^j A_j(\alpha_i) = 0$, where $\deg(A_j) \leq D$, $\deg(A_0) \leq D + k$.
2. Find all polynomials $P \in \mathbb{F}_q^{<k}[x]$ (polynomials of degree at most k) such that:

$$\Lambda_P(x) \triangleq A_0(x) + \sum_{j=1}^{r} P(\lambda^{j-1}x)A_j(x) \equiv 0. \tag{1}$$

We must check the following:

Claim 3. *$A_0, \ldots A_r$ exist.*

Proof. The number of variables is $\geq (r+1)D + k$, since A_j can have degree up to D for $j \in [r]$ and A_0 can have degree up to $D + k$. If the number of variables is greater than the number of equations, which is n, then a solution exists. For this reason, we chose $D = \frac{n}{r+1} - \frac{k}{r+1}$. (Note this means the the lower bound on m from Claim 2 equals $D + k$). □

Claim 4. *If $m \geq D + k$ then $\Lambda_P(x) = 0$.*

Proof. We note that $\deg(\Lambda_p) \leq D+k$, since $A_0(x)$ has degree $\leq D+k$, $P(x)$ has degree $\leq k$, and $A_j(x)$ has degree $\leq D$ for all $j \in [r]$.

If $\forall j$, $P(\lambda^{j-1}\alpha_i) = \beta_i^j$, then $\Lambda_P(\alpha_i) = 0$. In other words, if the ith column has no errors in it, then Λ_P is zero in that column. This holds because $\Lambda_P(x) = A_0(x) + \sum_{j=1}^r P(\lambda^{j-1}x)A_j(x) = A_0(x) + \sum_{j=1}^r \beta_i^j A_j(x) = 0$ for $x = \alpha_i$, by the condition on the A_j.

□

Proof of Claim 2. Our goal is to find the coefficients of P. Say $P(x) = \sum c_i x^i$. Then:

$$P(\lambda^{j-1}x) = \sum c_i(\lambda^{j-1}x)^i = \sum c_i(\lambda^{j-1})^i x^i$$

So if $A_j(x) = \sum a_{ij}x^i$, then the coefficient of x^t in $\Lambda_P(x)$ is:

$$a_{0,t} + \sum_{j=1}^{r}\left(\sum_{l=0}^{t} c_l(\lambda^{j-1})^l \cdot a_{t-l,j}\right) \tag{2}$$

So $a_{0,t}$ is a linear form in terms of the coefficients c_l which we're interested in. Conveniently, these linear forms are naturally triangular: knowing the coefficient of x^0 tells us c_0, knowing c_0 and the coefficient of x^1 gives us c_1, and in general, the expression (2) gives that the coefficient of x^t is:

$$c_t \cdot \left(\sum_{j=1}^{r} a_{0,j} \cdot (\lambda^t)^{j-1}\right) + f(c_0, \ldots c_{t-1}) = 0$$

Here f is some linear function of the earlier coefficients c_l.

Rearranging this equation gives that c_t is determined by the earlier c_l, except in the case that $\sum_{j=1}^r a_{0,j}(\lambda^t)^{j-1} = 0$. To deal with this case, define the polynomial $B(Y) = \sum_{j=1}^r a_{0,j}Y^{j-1}$. Then the coefficient of c_t vanishes when $B(\lambda^t) = 0$. But this is a polynomial of degree at most r, so either B is identically zero, or B has at most r zeroes.

First, we show that we can assume B is not identically zero. Note that the coefficients of the B are the constant terms of $A_1, \ldots A_r$. Suppose all of these have zero constant term. Then A_0 must also have zero constant term, because $\Lambda_P(x)$ is identically zero. So we can just divide the entire set of A_j by x; this will give a set of polynomials still satisfying the desired properties and the desired degree bound. Repeat as needed until some A_j has a nonzero constant term; then we obtain B not identically 0.

Therefore, we can assume that B has at most r zeroes. Moreover, none of the λ^t are the same (they are distinct since λ has high order), so the set $B(\lambda^0), B(\lambda^1), \ldots B(\lambda^{k-1})$ contains at most r zeroes. Hence, there are at most r values of t for which we cannot determine the correct value of c_t, so in particular, we have list-decoded the message to within q^r possible codewords.

□

There is an extension of this problem to the following setting, which is important for many applications of this code: Given r-dimensional vectors $\beta_1, \ldots \beta_t$, with many possibilties for each column, find a polynomial which passes through some block on each one of these columns - or even k/rn of the columns. This is useful for a problem known as "list recovery": in particular, it gives that for all ε there exists q such that for all R, δ s.t. $R + \delta \leq 1 + \varepsilon$ there exist q-ary codes of rate R, list-decodable from δ fraction errors, for infinite many n.