Lecture 10 : Folded RS Codes, Local Decoding

Error-Correcting Codes (Spring 2016)
Rutgers University
Swastik Kopparty
Scribes: Harsha Tirumala & Amey Bhangale

1 Overview

In this lecture, we will continue studying the list decoding algorithm for folded Reed-Solomon codes that we saw in the previous lecture. After that we will briefly go over two tricks on top of Folded Reed-Solomon codes that improve the list decoding parameters.

In the second part, we will define what *local decoding* is and will give an algorithm for local decoding of Hadamard code and Reed-Muller codes.

2 Folded Reed Solomon codes

2.1 Encoding Review

Definition 1. *Let $\gamma \in \mathbb{F}_q^*$ be a generator of the multiplicative group of $\mathbb{F}_q$, and let $s \geq 1$ be an integer parameter where s divides $q-1$. Then a Folded Reed Solomon code (FRS) is a code, C, with alphabet $\Sigma = \mathbb{F}_q^s$, evaluation set $S = \{\gamma^{0.s}, \gamma^{1.s}, \ldots, \gamma^{(((q-1)/s)-1)s}\}$ and $C \subseteq \Sigma^{(q-1)/s}$ is given by*

$$C := \{(f(\gamma^{is+0}), f(\gamma^{is+1}), \ldots, f(\gamma^{is+s-1}))_{i=0}^{(q-1)/(s-1)} : f \in \mathbb{F}_q[X], deg(f) \leq d\}$$

example codewords: $c_1 = (f(\gamma^0), f(\gamma^1), \ldots, f(\gamma^{s-1}))$; $c_2 = (f(\gamma^s), f(\gamma^{s+1}), \ldots, f(\gamma^{2s-1}))$

- length of code $= \frac{q-1}{s} = |S| = n$
- alphabet $\Sigma = \mathbb{F}_q^s$
- Rate $= \frac{d}{q-1} = R \quad \Rightarrow d = R(q-1) = R.n.s$

2.2 List decoding algorithm for FRS codes

Suppose we are given a message r. Although r consists of a $(q-1)/s$-tuple of elements of $\mathbb{F}_q^s$, for convenience we will view it instead as a function from $\{\gamma^0, \gamma^1, \ldots, \gamma^{q-s-1}\}$ to $\mathbb{F}_q^s$ (where $r(\gamma^{is})$ is simply the index i coordinate of the received tuple).
The FRS list decoding algorithm is very similar to that for Reed Solomon codes:

- **Step 1** Find a polynomial $Q(X, Y_0, Y_1, \ldots, Y_{s-1})$ with $Q(\gamma^{is}, r((\gamma^{is})) = 0$ for all $i \leq (q-1)/s$

- **Step 2** If $P(X) \in \mathbb{F}_q[X]$ is such that its codeword is close to r, then we consider $h(X) = Q(X, P(X), P(\gamma X), \ldots P(\gamma^{s-1}X))$

  By construction of Q, if $r(\gamma^{is}) = (P(\gamma^{is}), P(\gamma^{is+1}), P(\gamma^{is+2}), \ldots, P(\gamma^{is+s-1}))$, then $h(\gamma^{is}) = 0$. Therefore, if r is close to the codeword of P, then h has many roots, which will imply $h(X) \equiv 0$ (since h is of low degree).

- **Step 3** We therefore will obtain that $Q(X, P(X), P(\gamma X), \ldots P(\gamma^{s-1}X))) \equiv 0$ whenever P is close to r. Now we have to solve for all such P and we will be done.

The first two steps of the algorithm were analyzed in the previous lecture. We will now show how to perform Step 3 in polynomial time.

2.3 Finding all candidate polynomials P

Given $Q(X, Y_0, Y_1 \ldots, Y_{s-1})$, the task is to find all the polynomials $P(X) \in \mathbb{F}_q[X]$ of degree $\leq d$ such that $Q(X, P(X), P(\gamma X), \ldots P(\gamma^{s-1}X)) = 0$
This task is non-trivial even for $s = 2$. We look at a simpler problem now (which our current problem eventually reduces to due to γ being a generator):

Given $Q(X, Y_0, Y_1)$, find all polynomials $P(X)$ of degree $\leq d$ such that $Q(X, P(X), P(X)^{100}) = 0$.
Let $\tilde{Q}(X, Y_0) = Q(X, Y_0, Y_0^{100})$. If $Q(X, P(X), P(X)^{100}) = 0$ then $\tilde{Q}(X, P(X)) = 0$
$\Rightarrow Y_0 - P(X) | \tilde{Q}(X, Y_0)$.

All that remains is to factorize $\tilde{Q}(X, P(X))$. This problem was easy because $P(X)^{100}$ is directly dependent on the output of $P(X)$(in general such a dependency may not arise). The upcoming magic trick, however, reveals such a dependency between $P(X), P(\gamma X)$.

2.3.1 Magic Trick - Significance of the generator element γ

Let $E(X) = X^{q-1} - \gamma$

Fact 2. *Let q be the characteristic of $\mathbb{F}$, $\alpha \in \mathbb{F}$ and $P(X) \in \mathbb{F}[X]$ then $P(\alpha^q) = P(\alpha)^q$.*[1]

Fact 3. *Since γ is a generator, $E(X)$ is irreducible.*

Since $E(X) = X^{q-1} - \gamma$, we have

$$X^{q-1} \equiv \gamma (\bmod E(X))$$

$$\Rightarrow X^q \equiv \gamma X (\bmod E(X))$$

$$\Rightarrow P(X^q) \equiv P(\gamma X) (\bmod E(X))$$

$$\Rightarrow P(X)^q \equiv P(\gamma X) (\bmod E(X)) \quad \text{(using Fact 2)}$$

[1] See Lemma 6 from : http://www.math.rutgers.edu/~sk1233/courses/finitefields-F13/intro.pdf

So if

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

then

$$Q(X, P(X), P(X)^q, P(X)^{q^2}, \ldots P(X)^{q^{s-1}}) \equiv 0(\ \texttt{mod}\ E(X)) \tag{2}$$

Except the $\texttt{mod}\ E(x)$ condition this expression looks similar to the one in section 2.3. Rewriting as coefficients of $Y_0, Y_1, \ldots Y_{s-1}$:

$$Q(X, Y_0, Y_1 \ldots, Y_{s-1}) = \sum_{\vec{i}} c_{\vec{i}}(X) {Y_0}^{i_0} {Y_1}^{i_1} {Y_2}^{i_2} \ldots {Y_{s-1}}^{i_{s-1}}$$

Consider the reduction map $\phi : \mathbb{F}_q[X] \to \mathbb{F}_q[X]/E[X] \simeq \mathbb{F}_{q^{q-1}} =: \mathbb{K}$. Define $R(Y_0, Y_1 \ldots Y_{s-1}) = \sum_{\vec{i}} \phi(c_{\vec{i}}(X)) {Y_0}^{i_0} {Y_1}^{i_1} {Y_2}^{i_2} \ldots {Y_{s-1}}^{i_{s-1}} \in \mathbb{K}[Y_0, Y_1, \ldots Y_{s-1}]$

Let $\tilde{R}(Y_0) = R(Y_0, {Y_0}^q, {Y_0}^{q^2}, \ldots {Y_0}^{q^{s-1}}) \in \mathbb{K}[Y_0]$. Since $\mathbb{K}$ is a field, by the root finding property we can find all $\alpha \in \mathbb{K}$ satisfying $\tilde{R}(\alpha)$ in time $\texttt{poly}(\log|\mathbb{K}|, q, deg(\tilde{R})) = \texttt{poly}(q^s, deg(Q))$ where q is the characteristic of the field $\mathbb{K}$. Thus, we get a following claim:

Claim 4. *If $P(X)$ satisfies Equation(1) then $\phi(P(X))$ satisfies $\tilde{R}(\phi(P(X))) = 0$. So we find $\phi(P(X))$ among the α that we just found.*

Once we have found all $\phi(P(X))$, we can invert the map ϕ and get the polynomials $P(X)$. Some $P(X)$ may not lie within the list decoding radius but that can be checked in polynomial time. The important point is all $P(X)$ that lie within the list decoding radius will be in the collection of polynomials that we get after applying the inversion map.

The following caveats may arise in this process:

1. $R(Y_0, Y_1, \ldots Y_{s-1}) = 0$. But then $E(X)|Q(X, Y_0, Y_1 \ldots Y_{s-1})$. So we can handle this case by replacing $Q(X, Y_0, Y_1 \ldots Y_{s-1})$ with $Q(X, Y_0, Y_1 \ldots Y_{s-1})/E(X)$ and continue with the same process of root finding described above..
2. $R(Y_0, Y_1, \ldots Y_{s-1}) \neq 0$ but $\tilde{R}(Y_0, Y_1 \ldots Y_{s-1}) = 0$. This could happen but we will rule it out by taking $deg_{Y_i}(Q) < q$. $\Rightarrow$ each monomial of R leads to a unique monomial in $\tilde{R}$.

Finally since $d < q$ we can convert Q to find $P(X)$ with $deg \leq d$; when we already know $\phi(P(X))$. Note that $R \neq 0 \Rightarrow \tilde{R} \neq 0$

2.4 Almost final List decoding algorithm

1. Interpolate $Q(X, Y_0, Y_1 \ldots Y_{s-1})$ with $(1, d, d \ldots d)$ weighted degree $\leq D$ such that Q vanishes with multiplicities M at each point $(a, r(a))$ where $a \in S$. This is possible if degrees of freedom for Q are greater than the vanishing constraints being imposed, i.e. if:

$$\approx \frac{D^{s+1}}{d^s.(s+1)!} > \frac{M^{s+1}}{(s+1)!}.n$$

$$\frac{D}{M} > (d^s.n)^{\frac{1}{s+1}} = ((sRn)^s.n)^{\frac{1}{s+1}} = s^{\frac{s}{s+1}} R^{\frac{s}{s+1}} n$$

2. Take $P(X) = \mathbb{F}_q[X]$ of deg $\leq d$ such that codeword of P has $\geq A$ agreements with r. If $r(a) = (P(a), P(\gamma a), P(\gamma^2 a) \dots P(\gamma^{s-1} a))$, then $h(X) = Q(X, P(X) \dots P(\gamma^{s-1} a))$ vanishes at a with multiplicity $\geq M$. So, $h(X)$ has $\geq AM$ total zeroes (including multiplicities). Since $deg(h) \leq D \Rightarrow$ If $D < AM$, then $h(X) \equiv 0$.
   $h(X) \equiv 0 \Rightarrow P$ satisfies the functional equation 1, so we can find it.
   (works as long as $A > \frac{D}{M} > (d^s n)^{\frac{1}{s+1}}$

2.5 Removing $s^{\frac{s}{s+1}}$ factor - Final Trick

Take the folded RS codes with folding parameters s and decode using t variate interpolation instead of $s+1$ variate for some $1 << t << s$ (eg : $t = \sqrt{s}$)

Remembering things

$(P(\gamma^0), P(\gamma^1), \dots P(\gamma^{s-1})) \parallel (P(\gamma^s), P(\gamma^{s+1} \dots P(\gamma^{2s}))) \parallel \dots \parallel (P(\gamma^{(s-1)s}), P(\gamma^{(s-1)s+1}), \dots P(\gamma^{s^2-1}))$
$r(a) = r_0(a) \quad \parallel \quad r_1(a) \quad \parallel \quad r_2(a) \quad \parallel \quad \dots r_{s-1}(a)$

Take sliding windows of size t i.e. re-bundle the received word $\in \Sigma^{\frac{q-1}{s}} = \mathbb{F}_q^{s\frac{q-1}{s}}$ to another received word $\in (\mathbb{F}_q^t)^{(s-t+1)\frac{q-1}{s}}$ where new length $n' = (s-t+1)\frac{q}{s}$.

If the original received word had A agreements, the new received word has $\geq (s-t+1)A$ agreements. Run the previous list decoding algorithm. This finds all $P(X)$ of deg $\leq d$ such that agreement of the codeword of $P, r \geq (d^t n)^{\frac{1}{t+1}}$.
(folding parameter t , evaluation set $= S.\{1, \gamma, \gamma^2, \dots \gamma^{s-t}\}$).
If $A.(s-t+1) \geq (d^t.(s-t+1).\frac{q-1}{s})^{\frac{1}{t+1}}$ then $P(X)$ will be found, i.e. if

$$A > \frac{1}{s-t+1}(d^t.(s-t+1).n)^{\frac{1}{t+1}}$$
$$A > \frac{1}{s-t+1}(R^t s^t n^t (s-t+1)n)^{\frac{1}{t+1}}$$
$$A > \frac{1}{s-t+1} R^{\frac{t}{t+1}} (s^t n^t (s-t+1))^{\frac{1}{t+1}} n$$
$$A > (R+\epsilon)n$$

By choosing $s >> 1, t = \sqrt{s}$ this gives a list decoding algorithm from $1 - R - \epsilon$ fraction errors with lists in time $\texttt{poly}(q^s) = n^{\texttt{poly}(\frac{1}{\epsilon})}$ with list size $= n^{\texttt{poly}(\frac{1}{\epsilon})}$ and alphabet size $n^{\texttt{poly}(\frac{1}{\epsilon})}$

3 Local Decoding

Let's start with a basic question about complexity of decoding a codeword; How much *time* do we need to decode a code $C \subseteq \Sigma^n$? - at least $\log_\Sigma |C|$ (size of the output)

Till now what we saw is how to encode a message of length k into a codeword of length n. If a received word is not very far from a correct codeword then we saw decoding algorithms which decode the received word to the closest codeword. Finally from the encoding map, we can recover the original message.

But now suppose, instead of knowing the complete message we are only interested in the message restricted to a few locations. This is exactly the setting of local decoding. More formally, the local decoding problem can be described as:

Local Decoding: Let $C \subseteq \{0,1\}^n$ be a code and the encoding map be $E : \{0,1\}^k \to C$. Given $r \in \{0,1\}^n$ and $i \in [k]$ such that $\Delta(r, E(m)) < \rho$ for some $m \in \{0,1\}^k$. Find m_i.

Since in this case the output size is just a bit (or *log* of the message alphabet size in general), the basic question is: do we need to run in time $\Omega(k)$ for this? and the answer to this question is NO!

Let's first define the model and want operations cost us. One can study the complexity of an algorithm in terms of the running time. The other measure of complexity of a decoding algorithm is the number of locations queried by the algorithm from the input/received word. We'll study the local decoding algorithms w.r.t. the later complexity measure.

Random access model: Accessing any given location in a received word is counted as one operation.

The decoder algorithm will be randomized and we would like the failure probability to be a small constant, say 0.01.

Let's define a class of locally decodable codes based on the number of locations an algorithm needs to access in order to recover any given location of a message and a fraction of errors that it can tolerate.

Definition 5 ((t,ϵ)-locally decodable code)**.** *$C \subseteq \Sigma^n$, with an encoding map $E : \Sigma^k \to \Sigma^n$, is (t,ϵ)-locally decodable code if there exists an algorithm A such that for all messages $x \in \Sigma^k$, $r \in \Sigma^n$ and $i \in [k]$ such that $\Delta(r, E(x)) \leq \epsilon$,*

$$\Pr[A(i,r) \neq x_i] \leq 0.01$$

and A accesses at most t co-ordinates from r.

Designing a code that supports local decoding is part of the problem: Not every code with good distance property supports a local decoding Ex. Reed-Solomon codes with d=n/2; One has to read at least $n/2$ locations of a received word to decode a message symbol at a particular location. (exercise!)

3.1 Local decoding of Hadamard code

Recall that Hadamard code of length $n = 2^k$ is the set of truth tables or evaluations of all linear functions on k variables. For this particular local decoding algorithm to work, we only take non constant functions. More formally, the Hadamard code $C_H \subseteq \{0,1\}^n$ is

$$C_H = \{f : \mathbb{F}_2^k \to \mathbb{F}_2 \mid \exists a \in \mathbb{F}_2^k \setminus 0^k \text{ and } f(x) = \langle a, x \rangle \text{ for all } x\}$$

Thus, $|C_H| = 2^k - 1$. Given a message $a \in \{0,1\}^k \setminus 0^k$ of length k, the encoding is given by the linear function $l_a : \mathbb{F}_2^k \to \mathbb{F}_2$ where $l_a(x) = \langle a, x \rangle$. As we have seen earlier, the distance between any two codewords is exactly $1/2$.

In a local decoding context, we have given a function $r : \mathbb{F}_2^k \to \mathbb{F}_2$ (received word) and $i \in [k]$ such that $\Delta(r, l_a) \leq 0.01$ for some $a \in \mathbb{F}_2^k \setminus 0^k$. The task is to locally decode the i^{th} bit of the original message i.e the bit a_i in this case.

Algorithm:

1. Pick $x \in \mathbb{F}_2^k$ u.a.r.
2. Query r at x and $x + e_i$
3. Output $r(x) + r(x + e_i)$.

Analysis: First note that this algorithm outputs a correct answer with probability 1 if there is no error in the received codeword. The reason is, if $r = l_a$ for some a then $r(x) + r(x + e_i) = l_a(x) + l_a(x + e_i) = l_a(x) + l_a(x) + l_a(e_i) = l_a(e_i) = \langle a, e_i \rangle = a_i$ where in the second equality we used the fact that l_a is a linear function.

Let's analyze the error probability if $\Delta(r, l_a) \leq 0.01$. Since we sampled x u.a.r we have

$$\Pr[r(x) \neq l_a(x)] \leq 0.01$$

Also, the distribution on $x + e_i$ is uniform in $\mathbb{F}_2^k$ and hence we have

$$\Pr[r(x + e_i) \neq l_a(x + e_i)] \leq 0.01$$

Thus, using union bound we get,

$$\Pr[r(x) \neq l_a(x) \vee \Pr[r(x + e_i) \neq l_a(x + e_i)] \leq 0.02$$

and hence, with probability at least 0.98, both the queried locations are uncorrupted and when this event happens, we get $r(x) + r(x + e_i) = l_a(x) + l_a(x + e_i) = a_i$ as above.

Remark 6. *The Hadamard code can be locally decoded with* 2 *queries but the encoding size is exponential in the message length. It turns out that this cannot be avoided i.e. Every* 2 *query locally decodable code must have codeword of length* $2^{\Omega(k)}$ *where* k *is the message length [GKST02, KD03].*

3.2 Local Decoding of Reed-Muller code

In Reed Muller code, we look at the evaluations of m-variate degree d polynomials over $\mathbb{F}_q^m$. Notice that RM codes are generalization of Hadamard code where (in Hadamard code) the codewords are evaluations of linear (degree 1) functions over $\mathbb{F}_2$.

We will slightly modify the encoding map for RM codes that we saw in lecture 3[2]. We do this modification to ensure that message string stay as it is in some subset of locations of the correct encoding. For this, pick any subset $S \subseteq \mathbb{F}_q^m$ of size $\binom{m+d}{d}$ such that specifying values of a degree d polynomial at these points in S uniquely defines the polynomial. There are many such subsets (exercise!). From now onwards we let S be one of these subsets.

Given a message (evaluations on S) $g : S \to \mathbb{F}_q$, using the above argument, extend it to all $\mathbb{F}_q^m$. In other words, encoding of g is the evaluation of unique polynomial f of degree d such that $f_{|S} = g$.

In this section, we'll see how to locally decode RM code (for $d \leq q-2$) with the above encoding when the fraction of errors is at most say $1/100q$. In the next section, we will remove the dependence of q from the error bound.

Setup : Given $r : \mathbb{F}_q^m \to \mathbb{F}_q$ (received word) and $x \in S$ such that $\Delta(r, f) \leq \frac{1}{100q}$ where $\texttt{degree}(f) \leq d \leq q-2$ find $f(x)$.

Informally, the algorithm is as follows: We query the received word at bunch of points from $\mathbb{F}_q^m$ such that every query point is uniformly distributed in $\mathbb{F}_q^m$. The only place we use this uniformity of query points is to bound the probability that all the queried points are uncorrupted. Importantly, we choose the query points carefully so that the correct codeword restricted to the queried locations is evaluation of a low degree univariate polynomial, say $h(X)$, and the required message bit is $h(0)$. Thus, if all the queried locations are uncorrupted then we can interpolate the correct polynomial and thus can compute $h(0)$.

Algorithm:

1. Pick $\vec{y} \in \mathbb{F}_q^m$ u.a.r.
2. For each $\alpha \in \mathbb{F}_q^*$, let $\vec{z}_\alpha = \vec{x} + \alpha\vec{y}$.
3. Query all $r(\vec{z}_\alpha)$.
4. Let $U : \mathbb{F}_q^* \to \mathbb{F}_q$ be such that $U(\alpha) = r(\vec{z}_\alpha)$ for all $\alpha \in \mathbb{F}_q^*$.
5. Let $h(X)$ be the unique polynomial of degree at most d that agrees with U. If there does not exists such polynomial then output $fail$.
6. Output $h(0)$.

Remark 7. *The set $L = \{\vec{z}_\alpha = \vec{x} + \alpha\vec{y} \mid \alpha \in \mathbb{F}_q\}$ is a line in F_q^m passing through $\vec{x}$ in the direction $\vec{y}$. As we are picking the direction $\vec{y}$ (step 1) uniformly at random, step 3 of the algorithm is basically querying all but one point from a random line in F_q^m passing through $\vec{x}$. See figure 1.*

Analysis: We analyze the algorithm by stating a simple observation.

Observation 8. *For each $\alpha \in \mathbb{F}_q^*$, $\vec{z}_\alpha$ is uniformly distributed on $\mathbb{F}_q^m$.*

[2] `http://www.math.rutgers.edu/~sk1233/courses/codes-S16/lec3.pdf` (section 2)

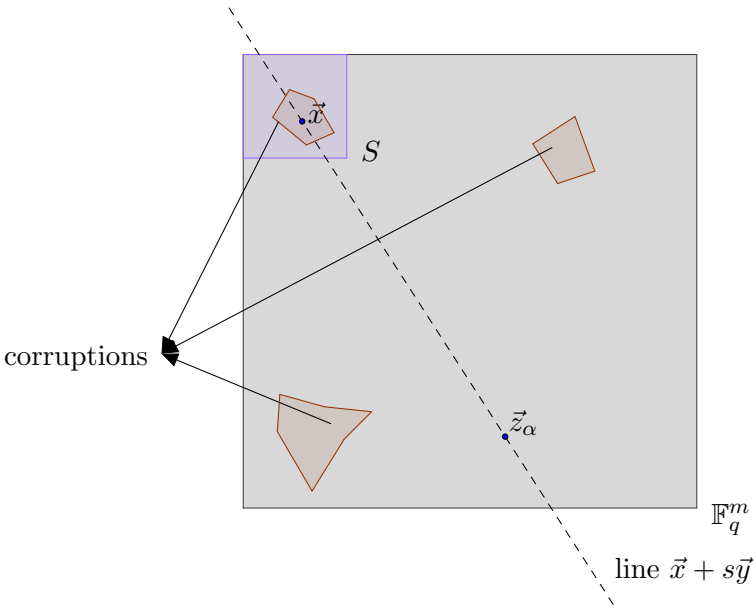

Figure 1: *Illustration of local decoding of RM code*

Given the condition that $\Delta(r, f) \leq \frac{1}{100q}$, for a fix $\alpha \in \mathbb{F}_q^*$, using observation 8, we have

$$\Pr[r(\vec{z}_\alpha) \neq f(\vec{z}_\alpha)] \leq \frac{1}{100q}.$$

Thus by union bound, we have

$$\Pr[\bigvee_{\alpha \in \mathbb{F}_q^*} r(\vec{z}_\alpha) \neq f(\vec{z}_\alpha)] \leq \frac{1}{100}.$$

Hence, with probability at least 0.99, all the queried locations are the correct encoded values. We will now proceed with the analysis given that this event happens.

We have, $U(s) = f(\vec{x} + s\vec{y})$ for all $s \in \mathbb{F}_q^*$. Note that $f(\vec{x} + s\vec{y})$ for fixed vectors $\vec{x}$ and $\vec{y}$ is an univariate degree d polynomial in a variable s. Thus, U is the evaluation of degree d polynomial on $\mathbb{F}_q^*$. Since $d \leq q - 2$, these evaluations uniquely define a degree d polynomial. Hence, step 5 of the algorithm succeeds and we have $h(s) = f(\vec{x} + s\vec{y})$ for all $s \in \mathbb{F}_q$. Substituting $s = 0$ gives $h(0) = f(\vec{x})$ as required.

Local decoding with constantly many queries: Let us fix the parameters from the above local decoding algorithm. Let's set $q = c$, a constant and $d = q - 2$. Then, the code parameters are:

1. Length of a code, $n = q^m$
2. Length of message, $k = |S|$ (number of m-variate monomials of degree at most d) $= \binom{d+m}{d} \approx \frac{m^d}{d!} = \frac{m^{q-2}}{(q-2)!}$

Thus, $n = 2^{O(k^{\frac{1}{q-2}})} = 2^{O(k^{\frac{1}{c-2}})}$ for a constant c. Also, from the above algorithm the query complexity is $t = q - 1 = c - 1$ a constant. Hence, for these setting of parameters, we get a code whose length is exponential in the message length (rate is not a constant) but it can be locally decoded using only constantly many queries from a constant fraction of error!

Remark 9. *For a general $t > 2$ (a constant) number of queries, $n \geq k^{1+O(1/t)}$ [Woo07] i.e. a locally decodable code with a constant number of queries cannot have a constant rate!*

Big open question - Can we have $t = O(\log k)$ query locally decodable code with constant rate?

The important part of the analysis is to recover the univariate polynomial $f(\vec{x} + s\vec{y})$ (for any $\vec{y}$) by querying few locations. In the above analysis, we argued that if all the locations queried are uncorrupted then we can recover the univariate polynomial. For this event to happen with constant probability, we needed that the fraction of errors is at most $O(1/q)$. But this problem of recovering a polynomial from its evaluation, with possibly few corruptions, is exactly the same as Reed-Solomon decoding. From the decoding algorithm of RS codes, we know that even if few locations corresponding to evaluation of a univariate polynomial are corrupted, we can recover the polynomial efficiently (Berlekamp-Welch algorithm). Thus, we will incorporate this fact in the above algorithm and get an improved (in terms of error fraction) local decoding algorithm for Reed-Muller codes.

3.3 Improved Local Decoding of RM Codes

Here again the setup is the same as above except the fact that we will set $d \leq q/2$ in order to tolerate constant fraction error in Reed-Solomon decoding as briefly described above.

Setup : Given $r : \mathbb{F}_q^m \to \mathbb{F}_q$ (received word) and $x \in S$ such that $\Delta(r, f) \leq \frac{1}{100}$ where $\texttt{degree}(f) \leq d \leq \frac{q}{2}$ find $f(x)$.

Algorithm:

1. Pick $\vec{y} \in \mathbb{F}_q^m$ u.a.r.
2. For each $\alpha \in \mathbb{F}_q^*$, let $\vec{z}_\alpha = \vec{x} + \alpha\vec{y}$.
3. Query all $r(\vec{z}_\alpha)$.
4. Let $U : \mathbb{F}_q^* \to \mathbb{F}_q$ be such that $U(\alpha) = r(\vec{z}_\alpha)$ for all $\alpha \in \mathbb{F}_q^*$.
5. Find a polynomial $h(X) \in \mathbb{F}_q[X]$ of degree at most d such that $\Delta_{\mathbb{F}_q^*}(h, U) < 1/4$ using Berlekamp-Welch algorithm. If there does not exists such polynomial then output $fail$.
6. Output $h(0)$.

Analysis: For any fixed $\alpha \in \mathbb{F}_q^*$, using observation 8, we have

$$\Pr[r(\vec{z}_\alpha) \neq f(\vec{z}_\alpha)] \leq \frac{1}{100}.$$

Thus in expectation, the number of queried locations that are corrupted is

$$\mathbb{E}[\text{number of corruptions on a line}] \leq \frac{q-1}{100}$$

By Markov's inequality,

$$\Pr[\text{number of corruptions } \geq 25\frac{q-1}{100}] \leq \frac{1}{25}.$$

Thus, with probability at least $24/25$, out of $q-1$ queried points on a line, at most $1/4$ th fraction of the locations are corrupted. Since $d \leq q/2$, the fraction of corrupted locations on a line is strictly less than half the minimum distance. Hence, Berlekamp-Welch algorithm decodes the polynomial $f(\vec{x}+s\vec{y}) =: h(s)$ correctly. Therefore, $h(0) = f(\vec{x})$ as required.

Locally decodable code with constant rate and sublinear query complexity: Fix $m = \Theta(1)$ to be a constant and $q = n^{1/m}$ where n is growing. Set $d = q/2$. We have

1. Message length, $k = \binom{d+m}{m} \approx \frac{d^m}{m!}$.
2. Codeword length $= n$.

Thus, rate of the code is $\frac{k}{n} = \frac{d^m}{m!}\frac{1}{q^m} = \frac{1}{m!}\frac{1}{2^m} = \Theta(1)$. With these settings, we get a constant rate code which is locally decodable from a constant fraction error with query complexity $t = q - 1 \approx n^{1/m} = O(k^{1/m})$ which is sublinear in the message length!

In the subsequent lectures, we will see following improvements:

1. A constant query locally decodable code where the codeword length is subexponential in the message length $n = 2^{2^{\sqrt{\log k}}}$.
2. Constant rate $R = \epsilon$ and query complexity $t = k^{(1-\delta)}$ where ϵ, δ are independent constants (unlike the above settings of parameters in improved RM local decoding) and k is the message length.

References

[GKST02] O. Goldreich, H. Karloff, L. J. Schulman, and L. Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. In *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 143–151, 2002.

[KD03] Iordanis Kerenidis and Ronald DeWolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 106–115. ACM, 2003.

[Woo07] David Woodruff. New lower bounds for general locally decodable codes. In *ECCC*, 2007.