Information and Coding Theory **Winter 2025**

Lecture 16: February 27, 2025

Lecturer: Madhur Tulsiani

1 List-decoding of Reed-Solomon codes

The decoding algorithm in the previous lecture requires the number of errors to be at most $\lfloor \frac{n-k}{2} \rfloor$, i.e. it requires error rate to be less than roughly $\frac{1}{2}(1 - \frac{k}{n}) \leq \frac{1}{2}$. Of course $1/2$ is a bound on the error rate (in the Hamming model) for *any code*, since the number of errors can be at most half the distance.

The notion of list-decoding allows us to toterate more errors, at the cost of producing a (short) list of multiple codewords when it is not possible to decide on a unique closest codeword. We will describe the algorithm by Sudan [Sud97], which list-decodes Reed-Solomon codes up to error rate $1 - 2\sqrt{k/n}$. For an detailed discussion of several results on list decoding, see the excellent survey by Guruswami [Gur07].

We can view the list decoding algorithm below as a generalization of the unique decoding algorithm discussed in the previous lecture. For unique decoding (from t errors), we found polynomials g and e with degrees $k - 1 + t$ and t respectively, such that

$$y_i \cdot e(a_i) \;=\; g(a_i) \qquad \forall i \in [n]\,,$$

where $a_1, \ldots, a_n$ are the evaluation points defining the code, and $y_1, \ldots, y_n$ are the (possibly corrupted) received values. This can be seen as finding a curve $h(X,Y)$ with $\deg_Y(h) = 1$, which passes through the points (a_i, y_i) for all $i \in [n]$. For $h(X,Y) = Y \cdot e(X) - g(X)$, we proved that $Y - f(X)$ must be a factor of $h(X,Y)$, where $f(X)$ is the polynomial defining the closest codeword.

In the case of list decoding, we still find a polynomial $h(X,Y)$passing through all the points (a_i, y_i), but allow a larger degree for Y. We will show that for any polynomial f in the desired error radius, $Y - f(X)$ must be a factor of $h(X,Y)$. We define the algorithm below, in terms degree parameters d_X and d_Y to be chosen later. Also, note that the algorithm requires computing all factors of $h(X,Y)$ of the form $Y - f(X)$. This can be done efficiently (in time poly(q)) though we do not discuss the details here. See Guruswami's survey for details of this step [Gur07].

List-decoding for Reed-Solomon codes

Input: $\{(a_i, y_i)\}_{i=1,\ldots,n}$

Parameters: $d_X, d_Y, t \in \mathbb{N}$

1. Find nonzero $h \in \mathbb{F}_q[X, Y]$ such that $\deg_X(h) \leq d_X$, $\deg_Y(h) \leq d_Y$ and $h(a_i, y_i) = 0$ for each $i \in [n]$.
2. Compute all factors of h that are of the form $Y - f(X)$.
3. Output all f from Step 2 such that $|\{i \in [n] \mid f(a_i) \neq y_i\}| \leq t$.

Lemma 1.1. *There exists $h(X, Y)$ that satisfies the conditions in Step 1 of the algorithm, if d_X, d_Y satisfy $(d_X + 1) \cdot (d_Y + 1) > n$.*

Proof: We observe that finding h is again equivalent to solving a system of linear equations. By writing $h(X, Y) = \sum_{0 \leq r \leq d_X} \sum_{0 \leq s < d_Y} c_{r,s} X^r Y^s$, the equation $h(a_i, y_i) = 0$ for $i \in [n]$ gives n linear equations in the coefficients $c_{r,s}$'s. Note that there are $(d_X + 1) \cdot (d_Y + 1)$ unknowns and n equations. Also, $c_{r,s} = 0$ for all r, s is a solution, since the system is homogeneous. Thus, if $(d_X + 1) \cdot (d_Y + 1) > n$, there exist multiple solutions and at least one of them must be nonzero. ■

Lemma 1.2. *Let $h \in \mathbb{F}_q[X, Y]$ be a polynomial that satisfies the conditions in Step 1 of the algorithm. Let $f \in \mathbb{F}_q[X]$ be a polynomial with degree at most $k - 1$, such that*

$$|\{i \in [n] \mid f(a_i) = b_i\}| \;\geq\; n - t \;>\; d_X + (k-1) \cdot d_Y\,.$$

Then, $(Y - f(X)) \mid h(X, Y)$ i.e., $Y - f(X)$ is a factor of $h(X, Y)$.

Proof: Let $I = \{i \in [n] \mid P(a_i) = y_i\}$. Then $h(a_i, f(a_i)) = 0$ for all $i \in I$. It follows that the univariate polynomial $h(X, f(X))$ has at least $|I|$ roots. But $h(X, f(X))$ has degree at most $d_X + (k-1) \cdot d_Y$. Since

$$|I| \;\geq\; n - t \;\geq\; d_X + (k-1) \cdot d_Y\,,$$

we must have $h(X, f(X)) \equiv 0$.

It follows that $(Y - f(X)) \mid h(X, Y)$. Indeed, we can write $h(X, Y) = (Y - f(X)) \cdot A(X, Y) + B(X, Y)$ where $\deg_Y(B) < \deg_Y(Y - f(X)) = 1$. So $B(X, Y)$ does not depend on Y. Now $h(X, f(X)) \equiv 0$ implies $B(X, Y) = B(X) \equiv 0$. ■

Choice of parameters. It remains to choose the values of the parameters d_X, d_Y and t to satisfy the conditions for the above lemmas. We can choose $d_X = \sqrt{n \cdot k}$ and $d_Y = \sqrt{n/k}$ and $t = n - 2\sqrt{n \cdot k}$, which satisfy the conditions above. Note that the list size is at most $d_Y = \sqrt{n/k}$. As an example, if $k = \varepsilon \cdot n$, we can tolerate an error rate of $1 - 2\sqrt{\varepsilon}$, while producing a list of size $\sqrt{1/\varepsilon}$.

Exercise 1.3. *Show that we can tolerate an even larger amount of error in the above algorithm, by using a more careful degree bound. Instead of the uniform bound* $\deg_X(h) \le d_X, \deg_Y(h) \le d_Y$*, we take* h *to be a sum over all monomials of the form* $X^r Y^s$ *such that* $r + (k-1) \cdot s < (n-t)$ *i.e., in a single monomial, the degree of* X *can even be as large as* $n - t - 1$*, if (say)* $s = 0$*. Show that we can now take correct* $t = n - \sqrt{2nk}$ *errors.*

1.1 A different definition of Reed-Solomon codes

We defined the encoding for Reed-Solomon codes as mapping coefficients for a polynomial to evaluations. Given $m_0, \ldots, m_{k-1} \in \mathbb{F}_q$, we defined

$$f(X) \;=\; m_0 + m_1 \cdot X + m_2 \cdot X^2 + \cdots + m_{k-1} \cdot X^{k-1}\,,$$

and defined, for a fixed $S = \{a_1, \ldots, a_n\} \subseteq \mathbb{F}_q$,

$$\mathsf{Enc}(m_0, \ldots, m_{k-1}) \;=\; (f(a_1), \ldots, f(a_n))\,.$$

However, by Lagrange interpolation, we can also specify a unique polynomial f of degree at most $k-1$, by specifying its values on k distinct inputs. Consider $H = \{a_1, \ldots, a_k\} \subset S$. We now think of the "message" in $\mathbb{F}_q^k$ as an arbitrary function $h : H \to \mathbb{F}_q$. We then define f to be the unique polynomial of degree at most $k-1$, consistent with these values. By Lagrange interpolation, we can write f as

$$f(X) \;=\; \sum_{i=1}^{k} h(a_i) \cdot \prod_{j \in [k] \setminus i} \left(\frac{X - a_i}{a_j - a_i} \right) \;=\; \sum_{i=1}^{k} h(a_i) \cdot \delta_{a_i}(X)\,.$$

where the polynomials $\delta_{a_i}(X)$ above are degree-$(k-1)$ polynomials satisfying $\delta_{a_i}(a_i) = 1$ and $\delta_{a_i}(a_j) = 0$ for all $j \in [k] \setminus i$. For f as defined above, we write

$$\mathsf{Enc}(h) \;=\; (f(a_1), \ldots, f(a_n))\,.$$

This encoding has the advantage that the message $(h(a_1), \ldots, h(a_k)) = (f(a_1), \ldots, f(a_k))$ is actually *contained* in the encoding. We will extend the above encoding to the case of Reed-Muller codes, and show that this allows for a very interesting notion of decoding which we call "local decoding".

Exercise 1.4. *Find the generator matrix for the above encoding, which maps* $h \in \mathbb{F}_q^k$*, to the codeword* $(f(a_1), \ldots, f(a_n))$ *as described above.*

2 Reed-Muller codes

One limitation of Reed-Solomon code is that it requires large field, in particular, $q \ge n$. Reed-Muller codes are generalization of Reed-Solomon codes that can be defined on any field size, even over $\mathbb{F}_2$. Specifically, the Reed-Muller code $\mathrm{RM}_q(d, m)$ is a linear code over $\mathbb{F}_q$, where the message $(c_{i_1,\ldots,i_m})_{0 \le i_1+\cdots+i_m \le d}$ is identified with the polynomial

$$f(X_1, \ldots, X_m) \;=\; \sum_{0 \le i_1+\cdots+i_m \le d} c_{i_1,\ldots,i_m} \cdot X_1^{i_1} \cdots X_m^{i_m} \,,$$

which is a multivariate polynomial of total degree at most d in m. The encoding maps the coefficients to $\{f(z_1, \ldots, z_m)\}_{z_1,\ldots,z_m \in \mathbb{F}_q}$, i.e. the codeword is the evaluation of f over all points in $\mathbb{F}_q^m$.

We will actually consider *subcode* of the Reed-Muller code, which has the property that the message is contained in the codeword, as we discussed for the alternate Reed-Solomon code above.

2.1 A subcode of the Reed-Muller code

Fix $H \subseteq \mathbb{F}_q$ such that $|H| = k$, and let $h : H^m \to \mathbb{F}_q$ be an arbitrary function. As in the case of Reed-Solomon codes, we will take the encoding of h to correspond to a low-degree polynomial, which agrees with h on its domain H^m. Concretely, we take

$$\begin{aligned} f(X_1, \ldots, X_m) \;&=\; \sum_{a_1,\ldots,a_m \in H} h(a_1, \ldots, a_m) \cdot \prod_{i=1}^{m} \delta_{a_i}(X_i) \\ &=\; \sum_{a_1,\ldots,a_m \in H} h(a_1, \ldots, a_m) \cdot \prod_{i=1}^{m} \left(\prod_{u \in H \setminus a_i} \left(\frac{X_i - a_i}{u - a_i} \right) \right) \end{aligned}$$

Note that $\deg_{X_i}(f) \le k - 1$ for each $i \in [m]$. We take the encoding of h to be

$$\mathsf{Enc}(h) \;=\; \{f(z_1, \ldots, z_m)\}_{z_1,\ldots,z_m \in \mathbb{F}_q} \,.$$

As in the case of (the alternate view of) Reed-Solomon codes, this encoding has the property that the message is contained in the encoding.

Exercise 2.1. *Check that the encoding above is linear in h. Conclude that the code*

$$C \;=\; \left\{ \mathsf{Enc}(h) \mid h : H^m \to \mathbb{F}_q \right\}$$

is a subspace.

The dimension of the above code equals the dimension of the space of functions $h : H^m \to \mathbb{F}_q$, which is k^m. The block-length of the code equals the number of evaluation points $(z_1, \ldots, z_m)$, which is q^m. Note that the code here not only has a bound on the total degree of the polynomial f, but also has the restriction that $\deg_{X_i} \le k - 1$ for each $i \in [m]$. It thus forms a subcode (subspace) of the Reed-Muller code $\mathrm{RM}_q(m \cdot (k-1), m)$ with total degree $d = m \cdot (k-1)$.

2.2 Distance of Reed-Muller Codes

A codeword of the Reed-Muller code is an evaluation of some polynomial $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ over all of $\mathbb{F}_q^m$. Also, since the codes we considered are linear, the distance equals the minimum weight of a non-zero codeword, which we denote as $\mathsf{wt}(f)$.

$$\mathsf{wt}(f) \;=\; \left\{ (z_1, \ldots, z_m) \in \mathbb{F}_q^m \mid f(z_1, \ldots, z_m) \neq 0 \right\} .$$

The weight of any non-zero polynomial (a polynomial which is not identically zero) can be understood using the following lemma. While this is usually referred to as the Schwartz-Zippel lemma, or the DeMillo-Lipton- Schwartz-Zippel lemma, it actually has a longer history as described in (Section 3.1 of) this article by Arvind et al. [AJMR19]. We refer to it as the polynomial identity lemma.

Lemma 2.2 (Polynomial Identity Lemma). *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a non-zero polynomial with total degree at most $d = c_1 \cdot (q-1) + c_2$ with $c_2 < q-1$, then*

$$\mathop{\mathbb{P}}_{z_1,\ldots,z_m} [f(z_1, \ldots, z_m) \neq 0] \;\geq\; \frac{1}{q^{c_1}} \cdot \left(1 - \frac{c_2}{q}\right) .$$

Note that the above lemma, gives

$$\mathsf{wt}(f) \;\geq\; \frac{q^m}{q^{c_1}} \cdot \left(1 - \frac{c_2}{q}\right) .$$

In the subcode considered in Section 2.1, we considered polynomials with $\deg_{X_i}(f) \leq k-1$ for each $i \in [m]$. In this special case of bounds on the individual degrees, the polynomial identity lemma has a simpler statement and simpler proof.

Lemma 2.3. *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a non-zero polynomial with $\deg_{X_i}(f) \leq d_i$ for each $i \in [m]$. Then,*

$$\mathop{\mathbb{P}}_{z_1,\ldots,z_m} [f(z_1, \ldots, z_m) \neq 0] \;\geq\; \prod_{i=1}^{m} \left(1 - \frac{d_i}{q}\right) .$$

Proof: We prove the statement by induction on the number of variables. The case $m = 1$ follows from the observation that a univariate non-zero polynomial with degree at most d, has at most d roots. By factoring out different powers of X_m, we can write $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ as

$$f(X_1, \ldots, X_m) \;=\; \sum_{j=0}^{d} g_j(X_1, \ldots, X_{m-1}) \cdot X_m^j ,$$

where $d \le d_m$ is the largest exponent j such that $g_j(X_1, \ldots, X_{m-1}) \not\equiv 0$. Using induction, we then get that

$$\begin{aligned}
&\mathop{\mathbb{P}}_{z_1,\ldots,z_m}[f(z_1,\ldots,z_m) \neq 0] \\
&\ge \mathop{\mathbb{P}}_{z_1,\ldots,z_m}\left[f(z_1,\ldots,z_m) \neq 0 \bigwedge g_d(z_1,\ldots,z_{m-1}) \neq 0\right] \\
&\ge \mathop{\mathbb{P}}_{z_1,\ldots,z_m}[g_d(z_1,\ldots,z_{m-1}) \neq 0] \cdot \mathop{\mathbb{P}}_{z_m}\left[\sum_{j=0}^{d} g_j(z_1,\ldots,z_{m-1}) \cdot z_m^j \neq 0 \mid g_d(z_1,\ldots z_{m-1} \neq 0)\right] \\
&\ge \prod_{i=1}^{m-1}\left(1 - \frac{d_i}{q}\right) \cdot \left(1 - \frac{d}{q}\right) \;\ge\; \prod_{i=1}^{m}\left(1 - \frac{d_i}{q}\right) .
\end{aligned}$$

■

Another special case, with a similar proof, is when the total degree d is smaller than $q - 1$.

Lemma 2.4. *Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a non-zero polynomial with total degree $d < q - 1$ Then,*

$$\mathop{\mathbb{P}}_{z_1,\ldots,z_m}[f(z_1,\ldots,z_m) \neq 0] \;\ge\; 1 - \frac{d}{q} .$$

Proof: As before, we use induction on the number of variables, and write

$$f(X_1,\ldots,X_m) \;=\; \sum_{j=0}^{d'} g_j(X_1,\ldots,X_{m-1}) \cdot X_m^j ,$$

where $d' \le d$ is the largest exponent j such that $g_j(X_1, \ldots, X_{m-1}) \not\equiv 0$. We can write the probability of f being 0 as (omitting the input variables in the expressions below)

$$\begin{aligned}
&\mathop{\mathbb{P}}_{z_1,\ldots,z_m}[f(z_1,\ldots,z_m) \;=\; 0] \\
&= \;\mathbb{P}[g_{d'} = 0] \cdot \mathbb{P}[f = 0 \mid g_{d'} = 0] \;+\; \mathbb{P}[g_{d'} \neq 0] \cdot \mathbb{P}[f = 0 \mid g_{d'} \neq 0] \\
&\le \;\left(\frac{d - d'}{q}\right) \cdot 1 \;+\; 1 \cdot \left(\frac{d'}{q}\right) \;=\; \frac{d}{q}
\end{aligned}$$

where we used induction, and the fact that the total degree of $g_{d'}$ is at most $d - d'$. ■

Exercise 2.5. *Prove the general polynomial identity lemma (Lemma 2.2) using induction on the number of variables.*

2.3 Local Correction of Reed-Muller codes

Let $\{f(z_1,\ldots,z_m)\}_{z_1,\ldots,z_m\in\mathbb{F}_q}$ be a Reed-Muller codeword and assume that α fraction of the codeword is corrupted and instead we observe $\{\widetilde{f}(z_1,\ldots,z_m)\}_{z_1,\ldots,z_m\in\mathbb{F}_q}$. Therefore, we have:

$$\mathop{\mathbb{P}}_{z_1,\ldots,z_m\in\mathbb{F}_q}\left[f(z_1,\ldots,z_m)=\widetilde{f}(z_1,\ldots,z_m)\right]\geq 1-\alpha$$

Decoding the codeword would correspond to recovering the values $f(z_1,\ldots,z_m)$ for all $z_1,\ldots,z_m\in H$. However, suppose we are only interested in the value at *one* point $(z_1,\ldots,z_m)$. Of course, decoding the full codeword would also give the value at the point of interest. However, the running time may be polynomial in q^m which is the length of the codeword.

Reed-Muller codes have the interesting property that for any point $(z_1,\ldots,z_m)$, we can recover the value $f(z_1,\ldots,z_m)$ (with high probability) in time $\mathrm{poly}(q,m)$. Note in particular that the dependence on m is polynomial instead of the exponential dependence we would get if we tried to recover the entire codeword. Also, we need to only to read the value of $\widetilde{f}$ at $O(q)$ randomly chosen points. Thus, we don't even read the entire received word. If he consider the subcode defined in Section 2.1 such that the message is contained in the codeword f, then we can also recover any position of the message this way.

Instead of stating a general result, we illustrate the technique via an example.

Local correction example. Let f be a codeword of the subcode considered in Section 2.1, and let $q\geq 5km$ (where $k=|H|$). By Lemma 2.3, we know that the distance is at least $\frac{4}{5}q^m$. Assume that $\alpha=\frac{1}{10}$ fraction of the codeword is corrupted. Given $z=(z_1,\ldots,z_m)$ we want to find the value $f(z_1,\ldots,z_m)$. Pick $y\in\mathbb{F}_q^m$ at random where $y=(y_1,\ldots,y_m)$ and define $\ell_y(t)=z+ty$ where $t\in\mathbb{F}_q$. Note that $\ell_y(0)=z$.

Consider the univariate polynomial $g_y(t)\in\mathbb{F}_q[t]$ defined as

$$g_y(t)\ =\ f(\ell_y(t))\ =\ f(z+t\cdot y)$$

Note that the degree of g_y is at most $(k-1)\cdot m$, and our goal is to find the value $g_y(0)$, where we are allowed to work with a randomly chosen y The idea of the decoding is that for most random y, we will end up with a univariate polynomial $g_y(t)$, where the amount of error is small enough that we can use Reed-Solomon decoding for univariate polynomials. Specifically, we have that for all $t\neq 0$

$$\mathop{\mathbb{P}}_{y}\left[\tilde{f}(z+t\cdot y)\neq f(z+t\cdot y)\right]\ \leq\ \frac{1}{10}\,.$$

Thus, we can write

$$\mathop{\mathbb{E}}_{y}\left[\left|\left\{t\in\mathbb{F}_q\setminus\{0\}\mid\tilde{f}(z+t\cdot y)\neq f(z+t\cdot y)\right\}\right|\right]\ \leq\ \frac{q-1}{10}\,,$$

which implies by Markov's inequality that

$$\mathop{\mathbb{P}}_{y}\left[\left|\left\{t \in \mathbb{F}_q \setminus \{0\} \mid \tilde{f}(z+t\cdot y) \neq f(z+t\cdot y)\right\}\right| \;\geq\; \frac{2(q-1)}{5}\right] \;\leq\; \frac{1}{4}\,.$$

Thus, we have that with probability at least $3/4$ over the choice of y, the value of $g_y(t)$ is correct in at least $3(q-1)/5$ positions. We can then use Reed-Solomon decoding to recover the polynomial $g_y(t)$ for a randomly chosen y, and return $g_y(0)$.

References

[AJMR19] Vikraman Arvind, Pushkar S. Joglekar, Partha Mukhopadhyay, and S. Raja. Randomized polynomial-time identity testing for noncommutative circuits. *Theory of Computing*, 15(7):1–36, 2019. URL: http://www.theoryofcomputing.org/articles/v015a007, doi:10.4086/toc.2019.v015a007. 5

[Gur07] Venkatesan Guruswami. Algorithmic results in list decoding. *Found. Trends Theor. Comput. Sci.*, 2(2):107–195, January 2007. URL: http://dx.doi.org/10.1561/0400000007, doi:10.1561/0400000007. 1

[Sud97] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. URL: http://dx.doi.org/10.1006/jcom.1997.0439, doi:10.1006/jcom.1997.0439. 1