Introduction to Coding Theory **CMU: Fall 2018**

Lecture 10: Improved List Decoding of Reed Solomon Codes

Lecturer: Venkatesan Guruswami *Scribe: Keerthana Gurushankar*

In the previous class, we saw an algorithm for list decoding $[n, k+1, n-k]_q$ Reed Solomon codes upto $n - \sqrt{2kn}$ errors by reducing it to the polynomial reconstruction problem. [Sud97] However, recall that the Johnson Bound states that a code of distance $k+1$ can be list decoded upto $n-\sqrt{kn}$ errors. Our goal today is to improve the algorithm so it can list decode upto $n - \sqrt{kn}$ errors and thus meet the Johnson Bound. We achieve this using polynomial reconstruction with the added consideration of roots of high multiplicity.

1 Sudan's Algorithm

We briefly recap the algorithm from last class for list decoding $[n, k+1, n-k]_q$ RS codes upto $n - \sqrt{2kn}$ errors.

Given a Reed Solomon code $C_{RS} = \{(p(\alpha_1), \ldots, p(\alpha_n)) \in \mathbb{F}^n \mid p \in \mathbb{F}_q[X], \deg p \leqslant k-1\}$, a vector $y \in \mathbb{F}_q$ to be decoded and an agreement parameter t such that y agreed with a codeword in C_{RS} at at least $t \geqslant \sqrt{2kn}$ indices, we find all polynomials $f \in \mathbb{F}[X]$ of degree $\leqslant k$ for which $f(\alpha_i) = y_i$ at at least t values of i. Each f corresponds to a codeword $(f(\alpha_1), \ldots, f(\alpha_n))$ in C_{RS} which agrees with y at at least t indices.

We use the following algorithm to find all such polynomials.

Algorithm I: Polynomial Reconstruction

INPUT: $\mathbb{F}$, n, k, t and n points $(\alpha_i, y_i) \in \mathbb{F}^2$
OUTPUT: All polynomials f with degree $\leqslant k$ such that $|\{i \mid f(\alpha_i) = y_i\}| \geqslant t$.

Step 0: Set parameter $D = \lfloor \sqrt{2kn} \rfloor$

Step 1: Find a non-zero bivariate polynomial $Q \in \mathbb{F}[X, Y]$ with $\deg_x(Q) + k \deg_y(Q) \leqslant D$ (i.e. having $(1, k)$-weighted degree $\leqslant D$) such that $Q(\alpha_i, y_i) = 0$ at all n points

Step 2: Find all linear factors $(y - f(x))$ of $Q(x, y)$ and include f in output list if $\deg f \leqslant k$ and $f(\alpha_i) = y_i$ at at least t points

---

In this algorithm, to allow t as small as possible, we aimed to increase the number of variables available in Step 1 (i.e. coeffecients of the bivariate polynomial Q). However, as a tradeoff, we needed the $(1, k)$-weighted degree of Q to be as small as possible in Step 2. By cleverly choosing degree constraints, we arrived at a solution for $t > \sqrt{2kn}$. However, while hoping to meet the Johnson Bound ($t > \sqrt{kn}$) through further such optimizations, we posit that we unfortunately cannot do much better with this algorithm without significantly altering the current framework.

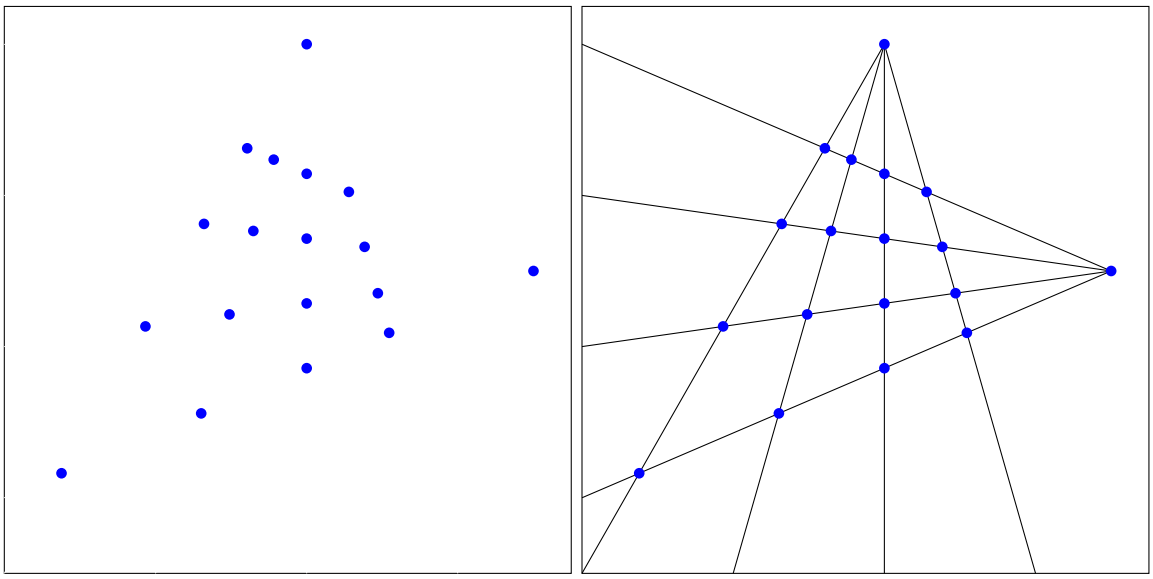

Fig. 1(a): Set of points input to the polynomial reconstruction problem;
Fig. 1(b): The correct answer to Fig. 1(a)

Consider for instance, the above pathologically constructed example as justification. We take $k = 1$ and $n = m^2 + 2$ for any m (m is taken to be 4 in the figure above), and try to find all polynomials (i.e. lines, since $k = 1$) passing through at least $m + 1$ of the $m^2 + 2$ points in Fig. 1(a). Clearly there are at least $2m$ such lines as pictured above. However the polynomial Q from Algorithm I, having degree $D \leqslant \sqrt{2kn} = \sqrt{2(m^2 + 2)}$ may have at most $\approx \sqrt{2}m$ linear factors and can thus cannot output all $2m$ lines. However, we note the self-intersecting nature of the curve in Fig. 1(b) and explore generalizing the polynomial reconstruction problem to one with roots of multiplicity specified to be possibly larger than 1.

2 Method of Multiplicity

The following result is an improvement of Algorithm I due to Guruswami and Sudan [GS98] which list decodes Reed Solomon codes upto the Johnson Bound.

Following our nose from the example above, we ask: how might we go about characterizing curves having high multiplicity roots? Borrowing from continuous math, we might like to define some notion of a derivative and suggest that the first $r - 1$ derivatives at a root of multiplicity r be zero. However differentiation does not carry very smoothly to finite fields. (For instance, the naively taken derivative of x^2 in $\mathbb{F}_2[X]$ would be $2x$, which is zero!).

So we must consider a different approach. Take a polynomial having a root of multiplicity r at 0. It must be divisible by x^r and thus have no monomials of degree $< r$. Indeed, this notion generalizes when combined with the translation of roots. And thus we have the following definition.

Definition 2.1 (Multiplicity of zeros) *A polynomial $Q(x, y)$ is said to have a zero of multiplicity r at the point $(\alpha, \beta) \in \mathbb{F}^2$ if the polynomial $Q^{(\alpha,\beta)}$ defined as $Q^{(\alpha,\beta)}(x, y) := Q(x + \alpha, y + \beta)$ has no monomials of total degree less than r.*

With this in mind, in Algorithm I, let's look for a bivariate polynomial Q having roots of multiplicity

r (to be specified) at each point (α_i, y_i) , rather than one simply passing through these points. We would be increasing the degree of Q, which is undesirable from the point of Step 1, where the additional constraints would force an increase in the degree of Q. However, in the Step 2, we will know that the polynomial f passes through not simply roots, but in fact singularities (high multiplicity roots) of Q. We would like to see how these trade-offs play off.

We thus sketch the following algorithm.

Algorithm II: Method of Multiplicity

INPUT: $\mathbb{F}$, n, k, and n points $(\alpha_i, y_i) \in \mathbb{F}^2$
OUTPUT: All polynomials f with degree $\leqslant k$ such that $|\{i \mid f(\alpha_i) = y_i\}| \geqslant t$.

Step 0: Set parameters for multiplicity r, and $\deg(Q)$ D /* to be specified later */

Step 1: Find a non-zero bivariate polynomial $Q \in \mathbb{F}[X, Y]$ of $(1, k)$-weighted degree $\leqslant D$ having roots of multiplicity r at each (α_i, y_i)

Step 2: Find all linear factors $(y - f(x))$ of $Q(x, y)$ and include f in output list if $\deg f \leqslant k$ and $f(\alpha_i) = y_i$ at at least t points

---

In the following lemma, we show that polynomial Q of the nature required in Step 1 exists when $\frac{D(D+2)}{2k} > n\binom{r+1}{2}$.

Lemma 2.2 *For any given parameters $n, r \in \mathbb{N}$ and n pairs $(\alpha_i, y_i) \in \mathbb{F}^2$, there exists a non-zero polynomial $Q(x, y)$ of $(1, k)$-weighted degree D with at least r zeroes at each of the n points (α_i, y_i) if $\frac{D(D+2)}{2k} > n\binom{r+1}{2}$.*

Proof: Each point (α, β) at which Q has a root of multiplicity r places the constraint that for all $i + j < r$, the coefficient $q_{i,j}^{(\alpha,\beta)}$ of $x^i y^j$ in $Q^{(\alpha,\beta)}$ is zero. Since $Q^{(\alpha,\beta)}(x, y) = Q(x + \alpha, y + \beta)$, the coefficients are given by

$$q_{i,j}^{(\alpha,\beta)} = \sum_{i'+kj' \leqslant D} \binom{i'}{i} \binom{j'}{j} q_{i',j'} \alpha^{i'-i} \beta^{j'-j}$$

Notice that this is a linear transformation of the coefficients $q_{i',j'}$ of Q. Thus each point generates $\binom{r+1}{2}$ homogenous linear equations as constraints. Thus, the n points (α_i, y_i) give us a system of $n\binom{r+1}{2}$ homogenous equations. The system has a non-trivial solution if the number of coefficients of Q is strictly greater. As proven in the previous lecture, the number of coefficients of Q is $\geqslant \frac{D(D+2)}{2k}$. Therefore a polynomial of the required nature exists if $\frac{D(D+2)}{2k} > n\binom{r+1}{2}$. ∎

We can also prove the following fact useful in Step 2.

Lemma 2.3 *For any input point (α_i, y_i), if $f \in \mathbb{F}[X]$ is a polynomial such that $f(\alpha_i) = y_i$ then $(x - \alpha_i)^r$ divides $Q(x, f(x))$.*

Proof: Since Q has a root of multiplicity r at (α_i, y_i), by the definition of multiplicity of roots, $Q^{(\alpha_i,y_i)}$ has no monomials of total degree $< r$, i.e. $Q^{(\alpha_i,y_i)}(x, y) = \sum_{l+m \geqslant r} q_{l,m}^{(\alpha_i,y_i)} x^l y^m$. We can

write Q in terms of $Q^{(\alpha_i,y_i)}$ as $Q(x,y) = Q^{(\alpha_i,y_i)}(x-\alpha_i, y-y_i)$. Then

$$\begin{aligned} Q(x,f(x)) &= Q^{(\alpha_i,y_i)}\left(x-\alpha_i, f(x)-f(\alpha_i)\right) \\ &= \sum_{l+m\geqslant r} q_{l,m}^{(\alpha_i,y_i)}(x-\alpha_i)^l(f(x)-f(\alpha_i))^m \end{aligned}$$

Since f is a polynomial, $(x-\alpha_i) \mid (f(x)-f(\alpha_i))$. It then follows from above that $(x-\alpha_i)^r \mid Q(x,f(x))$. ∎

The above can be used to prove the following lemma which lays out conditions for the correctness of Step 2 of our algorithm.

Lemma 2.4 *If a polynomial f has with degree $\leqslant k$, $f(\alpha_i) = y_i$ for at least t values of i and $rt > D$, then $y - f(x) \mid Q(x,y)$*

Proof: From Lemma 2.3, we have that $(x-\alpha_i)^r \mid Q(x,f(x))$ when $f(\alpha_i) = y_i$. Taking product over the t such values of i, the rt degree polynomial $\prod(x-\alpha_i)^r$ divides $Q(x,f(x))$ which has degree D. But in the case that $rt < D$, this must imply $Q(x,f(x)) \equiv 0$ and therefore $y - f(x) \mid Q(x,y)$. ∎

The lemmas we proved above combine to gives us conditions (for r, D) gives us sufficient conditions for Algorithm II to provide the necessary output. From Lemma 2.2, $\frac{D(D+1)}{2} > n\binom{r+1}{2}$ guarantees the successful completion of Step 1. Likewise from Lemma 2.4, $rt > D$ guarantees the completeness of Step 2. Thus we can arrive at a complete specification for our algorithm.

Algorithm II: Method of Multiplicity

INPUT: $\mathbb{F}$, n, k, and n points $(\alpha_i, y_i) \in \mathbb{F}^2$
OUTPUT: All polynomials f with degree $\leqslant k$ such that $|\{i \mid f(\alpha_i) = y_i\}| \geqslant t$.

Step 0: Compute parameters r and D such that $D < rt$ and $n\binom{r+1}{2} < \frac{(D+1)(D+2)}{2}$. To this end, we can in particular set

$$\begin{aligned} D &:= rt - 1 \\ r &:= 1 + \left\lfloor \frac{kn + \sqrt{k^2n^2 + 4(t^2-kn)}}{2(t^2-kn)} \right\rfloor \end{aligned}$$

Step 1: Find a non-zero bivariate polynomial $Q \in \mathbb{F}[X,Y]$ of $(1,k)$-weighted degree $\leqslant D$ having roots of multiplicity r at each (α_i, y_i)

Step 2: Find all linear factors $(y - f(x))$ of $Q(x,y)$ and include f in output list if deg $f \leqslant k$ and $f(\alpha_i) = y_i$ at at least t points

---

It can be shown similar to the analysis of Algorithm I, that Algorithm II can also be implemented efficiently.

But more importantly, we must note that Algorithm II meets the Johnson Bound. Lemma 2.2 gave us the constraint that $\frac{D(D+2)}{2k} > \frac{nr(r+1)}{2}$; and Lemma 2.4 gave that $rt > D$. Combining the two, we get the requirement that $rt(rt+2) > nkr(r+1)$. As we increase the value of r, the bound on t approaches $t > \sqrt{kn}$ and thus we can come arbitrrily close to the Johnson Bound.

3 Soft Decoding

A natural generalization of the method of multiplicities is to the case of weighted curve fitting: instead of assuming the same multiplicity r for all roots, we may weight each root with a multiplicity w_i representing a confidence value on the recieved symbol and placing more emphasis on smaller values. This connection has been explored by Koetter-Vardy [KV03], where it was shown how to choose weights optimally based on channel observations and transitions probabilities. It can be shown that the weighted polynomial reconstruction problem can be solved when $\sum w_i t > \sqrt{k \sum \binom{w_i+1}{2}}$.

References

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 28–37. IEEE, 1998.

[KV03] Ralf Koetter and Alexander Vardy. Algebraic soft-decision decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, 2003.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of complexity*, 13(1):180–193, 1997.