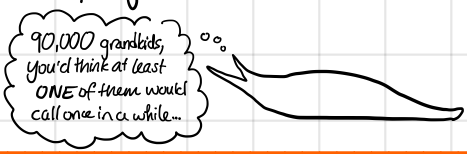
CS 250/EE387 – LECTURE 11 – GURUSWAMI-SUDAN ALG.

GASTROPOD FACT

A typical garden slug will have about 90,000 grandchildren – each slug lays 20-100 eggs several times per year.



AGENDA

0. Recap
1. Aside: How hard is decoding RS codes?
2. Sudan Algorithm
3. Guruswami-Sudan Algorithm.

0 RECAP.

Last time, we saw LIST-DECODING:

DEF. A code $\mathcal{C} \subseteq \Sigma^n$ is (p, L)-**LIST-DECODABLE** if $\forall y \in \Sigma^n$,

$$\left| \{ c \in \mathcal{C} : \delta(c, y) \leq p \} \right| \leq L.$$

The LIST-DECODING CAPACITY THM says that there $\exists$ codes that are $(p, 1/\epsilon)$-list-decodable with rate $R = 1 - H_2(p) - \epsilon$, for $p \leq 1 - 1/q$.

- That's the same trade-off as for random errors!
- Moreover, notice that p can get as big as $1 - 1/q$. If we demand unique decoding, the Plotkin bound says we can't hope to do better than $\frac{1-1/q}{2}$ with $R > 0$.

OUR NEXT QUESTION: How list-decodable are codes we know and love? For example, Reed-Solomon Codes?

Last time we saw the JOHNSON BOUND which says that codes with good distance are decently list-decodable.

- For RS codes, the Johnson bound says that an RS code of rate R is list-decodable up to distance $p \approx 1-\sqrt{R}$.

- Notice that list-decoding capacity is $p = H_q^{-1}(1-R) \approx 1-R$ for large q. So $p = 1-\sqrt{R}$ is less good than it could be.

① ASIDE: How hard is it to decode RS codes?

More precisely, given $w \in \mathbb{F}_q^n$, find $c \in RS_q(n,k)$ so that $\delta(w,c)$ is minimized.

How hard this depends a lot on the assumptions we can make about $\min_{c \in RS} \delta(w,c)$. For example, if $\exists c$ s.t. $\delta(c,w) < \frac{1-R}{2}$, then Welch-Berlekamp will do this in polynomial time. How about if $\delta(c,w)$ is larger?

Fraction of errors p	0 … $\frac{1-R}{2}$	$\frac{1-R}{2}$ … $1-\sqrt{R}$	$1-\sqrt{R}$ … $1-R$	$1-R$ … 1
List-decoding status	At most one codeword c with $\Delta(c,w) \le p$	By the Johnson Bound, there are $\le poly(n)$ codewords c with $\Delta(c,w) \le p$	$\left(\#c \text{ s.t. } \delta(c,w) \le p\right) \le ???$ For some choices of RS codes, it's exponential. For others, $O(1)$. In general, this is not well understood.	Definitely exponentially many codewords c s.t. $\delta(c,w) \le p$ for any p in this range. (B/c of list.dec.cap. thm)
How hard is it to find the closest codeword?	We can find c efficiently w/ (eg) Welch-Berlekamp or else efficiently decide no such c exists.	TODAY! We will see how to find all these $\le poly(n)$ codewords efficiently, and then we can search to find the closest.	Also ???. But there are some p's in here where doing MLD up to distance p is as hard as discrete log.	For large enough p, this is known to be NP-hard (you'll get a taste of this on your HW).

(2) SUDAN ALGORITHM

The Sudan Alg is a warmup to the GURUSWAMI-SUDAN alg, which will be able to efficiently list-decode RS codes up to the Johnson bound, $\rho = 1-\sqrt{R}$.

(2A) BIVARIATE POLYNOMIALS

A bivariate polynomial $Q(X,Y) \in \mathbb{F}_q[X,Y]$ is:

$$Q(X,Y) = \sum_{\substack{i=0,\dots,m_X \\ j=0,\dots,m_Y}} \alpha_{ij} X^i Y^j, \quad \text{where} \quad \begin{array}{l} m_X =: \deg_X(Q) \\ m_Y =: \deg_Y(Q) \end{array}$$

Notice that we can also think about Q as an element of $(\mathbb{F}_q[X])[Y]$:

$$Q(X,Y) = \sum_{j=0,\dots,m_Y} \underbrace{Q_j(X)}_{\text{The coefficients live in } \mathbb{F}_q[X]} \cdot Y^j$$

Polynomials in $(\mathbb{F}_q[X])[Y]$ behave a lot like a "normal" polynomial in Y.

FOR EXAMPLE: Consider $Q(Y) = Y^2 - 1$.
Then $Q(1) = 0$, which implies that $(Y-1) \mid Y^2 - 1$

Similarly, consider $Q(X,Y) = Y^2 - f(X)^2$.
Then $Q(X, f(X)) = 0$, which implies that $(Y - f(X)) \mid Q(X,Y)$

FACT. Let $Q(X,Y) \in \mathbb{F}_q[X,Y]$, and let $f \in \mathbb{F}_q[X]$. Then

$$Q(X, f(X)) \equiv 0 \iff (Y - f(X)) \mid Q(X,Y)$$

"$\equiv$" means "is identically 0" aka, all the coefficients are 0.

"divides." aka, $Q(X,Y) = (Y - f(X)) \cdot h(X,Y)$ for some $h \in \mathbb{F}[X,Y]$.

Moreover, we can find such f's efficiently, and there are at most $\deg_Y(Q)$ such f's.

(2B) RECALL the BERLEKAMP-WELCH ALGORITHM:

Given $y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$:

Recall, $E(X)$ was supposed to be the ERROR LOCATOR POLY, $E(X) = \prod_{i: y_i \neq c_i} (X - \alpha_i)$, so that $E(\alpha_i) \cdot f(\alpha_i) = E(\alpha_i) \cdot y_i \;\; \forall i$

1. Find low-degree polynomials $E(X), B(X)$ s.t. $E(\alpha_i) \cdot y_i = B(\alpha_i) \;\; \forall i = 1, \ldots, n$
2. Return $f(X) = B(X)/E(X)$

We can recast this in terms of bivariate polys:

1. Find $Q(X,Y)$ (meant to be $Q(X,Y) = E(X) \cdot Y - B(X)$) s.t. $Q(\alpha_i, y_i) = 0 \;\; \forall i = 1, \ldots, n$
2. Find a poly $f(X)$ s.t. $Q(X, f(X)) \equiv 0$, and return f.

(Notice that $f(X) = B(X)/E(X)$ will work in the Q we were supposed to find).

We'll use the same framework for SUDAN'S ALGORITHM for list-decoding.

PROBLEM: Given $y = (y_1, \ldots, y_n)$, k, and t, find all polynomials $f \in \mathbb{F}_q[X]$ s.t.:
- $\deg(f) < k$
- $f(\alpha_i) = y_i$ for at least t of the α_i's.

What t's can we handle? We'll see later!

(2C) Finally, SUDAN'S ALG.

In this context, Berlekamp-Welch is:

What exactly should this mean?

INTERPOLATION STEP 1. Find a LOW-DEGREE polynomial $Q(X,Y)$ so that $Q(\alpha_i, y_i) = 0 \;\; \forall i = 1, \ldots, n$

ROOT-FINDING STEP 2. Factor $Q(X,Y)$ to find polynomials $f(X)$ s.t. $Q(X, f(X)) \equiv 0$. Return all such f's.

- We can do STEP 1 as long as we have more variables (coeffs of Q) than constraints.
- To make sure that STEP 2 is correct, we'll have to argue that whenever $f(\alpha_i) = y_i$ for $\geq t$ values of i, then $Q(X, f(X)) \equiv 0$. The fact that the list is small will follow from the fact that Q is low-deg.

This algorithm basically works, and is called SUDAN'S ALGORITHM.

THM If $t > 2\sqrt{nk}$, then we can solve the list-decoding problem in polynomial time.

Before we prove the THM, we can ask how good this is.

$$(\#\text{agreements between } f \text{ and } y) = t > 2\sqrt{nk}$$

$$\Delta(f,y) = n - t < n - 2\sqrt{nk}$$

So this works up to radius $p \leq \frac{n-t}{n} = 1 - 2\sqrt{R}$.
Remember that we were shooting for $1-\sqrt{R}$, so this isn't quite right – but we'll get there!

Now we'll prove the THM, and finish specifying the alg. along the way.

pf/algorithm:

STEP 1 (INTERPOLATION). Choose $\ell = \sqrt{nk}$.

Find $Q(X,Y)$ s.t. $\deg_X(Q) \leq \ell$ and $\deg_Y(Q) \leq n/\ell$,
so that $Q(\alpha_i, y_i) = 0 \;\forall\, i = 1, \dots, n$.

To do this, we need:

$$\underbrace{(\#\text{coeffs in } Q)}_{(\ell+1)(\frac{n}{\ell}+1) \text{ of these}} > \underbrace{(\#\text{constraints})}_{n \text{ of these}}$$

and indeed we have $(\ell+1)(\frac{n}{\ell}+1) = n + \frac{n}{\ell} + \ell + 1 > n$ ✓

STEP 2 on NEXT PAGE

pf ctd.

STEP 2. (ROOT-FINDING STEP) Return all $f(X)$ s.t. $Q(X, f(X)) \equiv 0$.

Note that we can do this efficiently, and the size of our list will be at most $\deg_Y(Q) = n/\ell = n/\sqrt{kn} = 1/\sqrt{R}$, a constant.

Now we need to argue why this step is a good idea.

Suppose $\deg(f) < k$ and that $f(\alpha_i) = y_i$ for $\geq t$ vals of i.
We need to show that we will return f, so we need to show $Q(X, f(X)) \equiv 0$.

Let $R(X) := Q(X, f(X))$.
Then $\deg(R) \leq \deg_X(Q) + \deg(f) \cdot \deg_Y(Q) < \ell + k \cdot \frac{n}{\ell} = 2\sqrt{nk}$

This is why we chose $\ell = \sqrt{nk}$, to balance these two terms.

But $R(\alpha_i) = Q(\alpha_i, f(\alpha_i)) = Q(\alpha_i, y_i) = 0$
for at least t values of i.

So R has degree $< 2\sqrt{nk}$, but $t > 2\sqrt{nk}$ roots, hence $R(X) \equiv 0$, as desired.

3 GURUSWAMI–SUDAN ALG.

Now we'll fix this up so that we can actually get up to $p = 1 - \sqrt{R}$, meeting the JOHNSON BOUND.

TWO CHANGES:

1. We will change how we measure "LOW-DEGREE"
2. We will require something a bit stronger than $Q(\alpha_i, y_i) = 0$; we'll ask for Q to vanish with high MULTIPLICITY.

CHANGE 1.

DEF. The $(1,k)$-degree of $X^i Y^j$ is $i+kj$.
The $(1,k)$-degree of $Q(X,Y)$ is the max $(1,k)$-degree of any monomial in Q.

Just this change is enough to make SOME progress:

THM If $t > \sqrt{2nk}$, then we can solve the list-decoding problem in polynomial time.

pf. sketch Same alg, but now demand the $(1,k)$-degree of Q is $\leq \sqrt{2kn}$

STEP 1. INTERPOLATION.
Find $Q(X,Y)$ s.t. $(1,k)$-deg is $\leq \sqrt{2kn}$.

Turns out (FUN EXERCISE!) there are $> D^2/2k$ coeffs in a poly w/ $(1,k)$-deg $\leq D$. So

$$(\#\text{variables}) > \frac{(2kn)}{2k} = n = (\#\text{constraints})$$

and we can find Q.

STEP 2. ROOT-FINDING.
(same as before)

Now we have $\deg(R) = \deg(Q(X, f(X))) < (1,k)\text{-deg of } Q \leq \sqrt{2nk}$
So the argument goes through as before with a slightly better bound.

But we want $1-\sqrt{R}$! Not $1-\sqrt{2R}$!

CHANGE 2.

DEF. $Q(X,Y)$ has a root of multiplicity r at (a,b) if $Q(X+a, Y+b)$ has no terms of total degree $< r$.

Example: $Q(X,Y) = (X-1)^2(Y-1)$ has a root of multiplicity 3 at $(1,1)$, because $Q(X+1, Y+1) = X^2 \cdot Y$ which has total degree 3.

GURUSWAMI–SUDAN ALGORITHM.

Choose a parameter r

Suppose $t \geq \sqrt{kn(1+1/r)}$

1. INTERPOLATION STEP.

   Find a polynomial $Q(X,Y)$ with $(1,k)$-degree $D = \sqrt{kn \cdot r \cdot (r+1)}$ so that $Q(\alpha_i, y_i) = 0$ with multiplicity r for $i = 1, \ldots, n$.

2. ROOT-FINDING STEP.

   Return all f so that $Q(X, f(X)) \equiv 0$.

   [Notice that there are $\leq \deg_Y(Q) \leq D/k \approx r/\sqrt{R}$ of these.]

ANALYSIS:

Again we need to show that 1. is possible and that 2. is a good idea.

1. FUN EXERCISE: The number of constraints in "$Q(\alpha_i, y_i) = 0$ w/ mult. r" is $n \cdot \binom{r+1}{2}$.

   So that's MORE constraints than before, which seems like a bad thing....

   we'll see later why it's actually good.

   The number of variables is still $D^2/2k$, so we need

$$\frac{kn \cdot r \cdot (r+1)}{2k} \geq n \cdot \binom{r+1}{2} = \frac{n \cdot r \cdot (r+1)}{2}$$

   which is TRUE. ✓

2. Let $R(X) = Q(X, f(X))$ as before.

   Then not only does $R(X)$ have $\geq t$ roots [as before], it has $\geq t$ roots which EACH have multiplicity r.

Justification on next page...

CLAIM. If $f(\alpha_i)=y_i$, then $(X-\alpha_i)^r \mid R(X)$.

Pf. Let's drop the i subscripts for notational sanity.
Recall that since Q has a root of multiplicity r at (α, y),
$Q(X+\alpha, Y+y)$ has no terms of total degree $< r$.

Now, consider $\bar{f}(X) := f(X+\alpha)-y$. We have

$$R(X+\alpha) = Q(X+\alpha, f(X+\alpha)) = \underbrace{Q(X+\alpha, \bar{f}(X)+y)}_{\text{This is a sum of monomials } X^c \cdot \bar{f}(X)^d \text{ where } c+d \geq r.}$$

Now, since $f(\alpha)=y$, $\bar{f}(0)=0$, so $\bar{f}$ has no constant term.
Thus, those monomials $X^c \bar{f}(X)^d$ are all divisible by X^{c+d},
and hence are all divisible by X^r.

Then $X^r \mid R(X+\alpha)$, which means $(X-\alpha)^r \mid R(X)$,
as desired.

Now given this claim, the fact that $f(\alpha_i)=y_i$ for at least t different i's means that $R(X)$ has $t \cdot r$ roots, counting multiplicities.

This is why it was OK to take a hit in the number of constraints! Now we get $r \cdot t$ roots instead of r.

Since $\deg(R) \leq D$, if R is nonzero we must have

$$tr < D$$
$$\sqrt{kn(1+1/r)} \cdot r < \sqrt{kn\, r(r+1)}$$
$$\sqrt{kn\, r(r+1)} < \sqrt{kn\, r(r+1)} \quad \lightning$$

That's not true, so $R(X) \equiv 0$, and the proof concludes as before.

This proves the following theorem:

THM If $t > \sqrt{nk(1+1/r)}$ then we can solve the list-decoding problem in poly(n) time, with list size $r \cdot \sqrt{\frac{n}{k}}$.

Once again, we calculate that this means we can take $p = \frac{n-t}{n} = 1 - \sqrt{R(1+1/r)}$, so we conclude

THM. For all $r > 0$, RS codes of rate R are $\left(1-\sqrt{R(1+1/r)}, r/\sqrt{R}\right)$ List-decodable, and the Guruswami-Sudan algorithm can do the list-decoding in time poly(n,r).

Thus, we can rachet up r as large as we like (say, r=poly(n)) and approach the Johnson bound with polynomial-time algorithms. HOORAY!

The moral of the story:

WE CAN EFFICIENTLY LIST-DECODE RS CODES up to the JOHNSON BOUND

NOTE.
As presented, the Guruswami-Sudan algorithm runs in time $O(n^3)$, but people have optimized the heck out of it and it can be made to run in time $O(n\log(n))$.

disclaimer: maybe there's another loglog(n) factor in there, I forget.

QUESTIONS TO PONDER

1. What breaks in the GS algorithm beyond the Johnson bound?
2. Can you come up with a "bad" list of close-together RS codewords beyond the Johnson bound?
3. What if I modify the constraints so that instead of "$f(\alpha_i) = y_i$" they are "$f(\alpha_i) \in \{y_i, y_i', y_i''\}$"