Algebraic Methods in Computer Science 17.12.14

Lecture: 8

Lecturer: *Amir Shpilka* Scribe: *Roi Tabach*

In this lecture (and the next few) we will continue using the polynomial method to analyze algorithms for Error Correcting, as well as other problems. We will see a new method defining polynomials that vanish over some subsets, we will be able to give an intelligent bound on the polynomials degrees. In fact, we will start by finding in how many places the polynomial vanishes, and then try to force it's degree to be less then that. We will look into:

- Reed-Solomon Error Correcting Codes
- Berlekamp-Welch Alg. for Error Decoding Up To Half Distance.
- List Decoding
- Combinatorial Limit on the List Size in Reed-Solomon Codes.
- Kakeya's Problem - Real and in Finite Fields.
- Solution of Kakeya's Problem in Finite Fields.

Reed-Solomon Codes:

Until today, we had only codes of the form $C \subset \{0,1\}^n$. Now we will look into an Error Correcting Code over a field F_q. We are assuming $n \geq q$, and the message sent is of the form $(\alpha_1, .., \alpha_n) \in C$, where $C \subset F_q^n$. It's worth mentioning that the code's real length is $n \cdot \log q$, since it costs $\log q$ bits to send each α_i. And only n errors are enough for messing up all the α_is. So Reed-Solomon Codes are poor for randomly distributed errors. However, they are quite good for continuous errors (when the errors tend to come consecutively.), so they are interesting for some cases. Another application could be encoding the data in RS ECC, and then encoding each α_i in another code. And most importantly, they are beautiful, see motto in lecture 1.

Let $RS(n,k)_q$ denote the Reed-Solomon code over F_q^n. The messages are polynomials of degree k.

$$p = \sum_{i=0}^{k-1} a_i x^i \in F_q[x]$$

and they will be represented as k-tuples: $(a_0, a_1, \ldots, a_{k-1}) \in F_q^n$.

In order to Encode messages, instead of sending the coefficients of the polynomial (which is the natural way to encode it), we will send it's values. The intuition for this stems from the well known fact that in order to recreate a polynomial of degree $k-1$, it is enough to know k of it's values. So intuitively, we will send n values ($n > k$), and allow for erring in up to $\frac{n-k}{2}$ of them. If we would have been working in a scenario where data is lost, but not altered, we could have allowed up to $n-k$ losses. This fact is the core of the Reed-Solomon ECC.

We will choose in advance n values, $\alpha_1, \ldots, \alpha_n$, and define the following mapping:

$$f \mapsto (f(\alpha_1), \ldots, f(\alpha_n)) \in {F_q}^n$$

It is easy to see why $\mathrm{RS(n,k)_q}(f+g) = \mathrm{RS(n,k)_q}(f) + \mathrm{RS(n,k)_q}(g)$, and similarly why $\mathrm{RS(n,k)_q}(c \cdot f) = c \cdot \mathrm{RS(n,k)_q}(g)$. We have established the code is linear, in fact it's matrix is exactly:

$$\begin{pmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-1} \\ \vdots & \vdots & \diagdown & \vdots \\ 1 & \alpha_n & \ldots & \alpha_n^{k-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{k-1} \end{pmatrix} = \begin{pmatrix} f(\alpha_1) \\ \vdots \\ f(\alpha_n) \end{pmatrix}$$

It easily derives that the code's dimention is k. It's distance, as a linear code, is simply:

$$D := \min_f \mathrm{Dist}(\mathrm{RS(n,k)_q}(f), 0)$$

And using the fact that a polynomial f who's $\deg(f) < k$ has at most $k-1$ roots, we get that

$$\mathrm{Dist}(\mathrm{RS(n,k)_q}) \leq n - (k-1) = n - k + 1 := d$$

This is in fact a tight bound, since we can choose $f = (x-\alpha_1) \cdot \ldots \cdot (x - \alpha_{k-1})$. We will also notice that $d + k = n + 1$, which is as good as we can hope to get, from the Singleton bound.

Algorithm for error correcting:

We are given a vector of n values, say, $(y_1, \ldots, y_n) \in {F_q}^n$, and we know there exists a polynomial who agrees with $n-t$ of these values. We wish to find that polynomial, i.e. find $f \in F_q[x]$ so that $f(\alpha_i) \neq y_i$ at most t times.

Let S be the set of wrong indexes, and let E be the Error Locating Polynomial, i.e. $E := \prod_{i \in S}(x - \alpha_i)$. Obviously we do not know the value of E, but we can still use it. Let

$$N(x) := E(x) \cdot f(x) \tag{1}$$

a polynomial of degree $\leq t + k - 1$. We have defined two formal polynomials who are related to f and the errors. We are about to notice two important facts: the first is that $f = \frac{N}{E}$,

and the second is that we can have linear equations that will give us the coefficient of N and E.

So, we notice that for every α_i,

$$N(\alpha_i) = \begin{cases} 0 & i \in S \\ E(\alpha_i) \cdot y_i & i \notin S \end{cases} \tag{2}$$

And in fact $N(\alpha_i) = E(\alpha_i) \cdot y_i$ in both cases. These are the linear equations we will be solving. If we denote

$$N(x) := \sum_{i=0}^{t+k-1} b_i x^i$$

and

$$E(x) := \sum_{i=0}^{t} c_i x^i$$

Then each α_k gives us:

$$\left(\sum_{i=0}^{t+k-1} b_i \alpha_k^i \right) = \left(\sum_{i=0}^{t} c_i \alpha_k^i \right) \cdot y_i$$

There are n equations and $2 \cdot t + k + 1$ variables ($\vec{c}$ and $\vec{b}$), but we know there is a solution to it, because we know N,E exists. And they are a solution. (It derives from the fact we know this errored messege originated from an actual message.) So assuming we will find somehow N', E' that solve the equations we got, (obviously we have no guarantee that $N' = N$,$E' = E$...), We will output from the algorithm $\frac{N'}{E'}$! Now we only need to prove that:

Theorem 1 (Correctness of the Decoding Algorithm)**.** $\frac{N}{E} = \frac{N'}{E'}$

Proof. It is enough to show that $N'E = NE'$. By definition (2), we get for all i,

$$N(\alpha_i) = E(\alpha_i) \cdot y_i \tag{3}$$

On the other hand, due to the fact that they are a solution of the equation set, we can say that for all i,

$$N'(\alpha_i) = E'(\alpha_i) \cdot y_i \tag{4}$$

Simple arithmetics from (3) and (4) we get that for all i,

$$N(\alpha_i) \cdot E'(\alpha_i) = N'(\alpha_i) \cdot E(\alpha_i)$$

We achieved an equality between two polynomials, on n points. It will suffice to show that they are of degree less then n. We have demanded

$$t \leq \frac{n-k}{2}$$

And so we get $\deg(EN'), \deg(E'N) = 2t + k - 1 \leq n$ □

List Decoding:

So far we have only looked at the distance the code has - meaning how much errors can we allow before we can't distinguish two codewords. Now we will try and look in a more generalized sense of this distance. Given a word $w \in {F_q}^n$ we would like to return all the codewords who agree with w in at least A places. We will say a Code C is (A, l)-list.dec. if for every codeword w, there are at most l codewords who agree with it in at least A coordinates.

Note that if the minimal distance of the code is d, and the code length is n, so for

$$A > n - \frac{d}{2}$$

there is at most one codeword who agrees with w on at least A places. Otherwise we wouldn't have been able to decode. So every code of length n and distance$> d$ is trivially $(n - \frac{d}{2}, 1)$-list.dec.

We will now take a deeper look into $\mathrm{RS(n,k)_q}$'s list.dec. bounds, and provide an algorithm for it's list decoding.

Theorem 2 (bound for $\mathrm{RS(n,k)_q}$ list.dec.)**.** $\mathrm{RS(n,k)_q}$ *is* $(\sqrt{n \cdot k}, n)$*-list.dec*

Proof. First we will note that the claim isn't trivial

$$\sqrt{n \cdot k} \overset{\text{ave. inequallity}}{<} \frac{n+k}{2} = \frac{n-(n-k)}{2} = n - \frac{d}{2}$$

Now we assume there are m polynomials that have encoding that agree with w in at least A coordinates. We will be looking in a bipartite graph. On it's Left Hand Side, we will have n vertexes corresponding to the n coordinates of w, named $w_1, \ldots, w_n$. On it's Right Hand Side we will have m vertexes corresponding to the m polynomials who's encoding agree with w, named $f_1, \ldots, f_m$. For each polynomial f_i, we will add an edge (j, i) for every coordinate w_j that f_i agrees with. If there are more than A js, we will choose a subset of them.

Notice that we get a graph with $m \cdot A$ edges, and that for every $i_1, i_2 \in$ RHS, f_{i_1}, f_{i_2} can have only $k-1$ shared neighbors. That is due to the fact that two polynomials of degree $\leq k$ that are equal on k points, are the same polynomial. The graph is bipartite, and $K_{k,2}$ is not a subgraph.

We will now count the number of $K_{1,2}$s in our graph. If we count from the RHS:

$$\#K_{1,2} \leq \overbrace{\binom{m}{2}}^{\text{number of poly. pairs}} \cdot \overbrace{(k-1)}^{\text{each pair has at most } k-1 \text{ shared coordinates}} \tag{5}$$

And by counting from LHS:

$$\#K_{1,2} = \sum_{i=1}^{n} \binom{d_i}{2} \tag{6}$$

combining (5) and (6) we get

$$\sum_{i=1}^{n} \frac{d_i \cdot (d_i - 1)}{2} \leq \frac{m \cdot (m-1) \cdot (k-1)}{2}$$

$$\sum_{i=1}^{n} \frac{d_i^2 - d_i}{2} \leq \frac{m \cdot (m-1) \cdot (k-1)}{2}$$

$$\sum_{i=1}^{n} d_i^2 - \sum_{i=1}^{n} d_i \leq m \cdot (m-1) \cdot (k-1)$$

We now notice that from Cauchy Schwartz we get:

$$\frac{\left(\sum d_i\right)^2}{n} \leq \sum d_i^2 \tag{7}$$

So in total we have:

$$\frac{\left(\sum_{i=1}^{n} d_i\right)^2}{n} - \sum_{i=1}^{n} d_i \overset{(7)}{\leq} \sum_{i=1}^{n} d_i^2 - \sum_{i=1}^{n} d_i \leq m \cdot (m-1) \cdot (k-1)$$

Notice that $\sum d_i = m \cdot A$, as they are both ways of counting the edges.

$$\frac{(m \cdot A)^2}{n} - m \cdot A \leq m \cdot (m-1) \cdot (k-1)$$

$$\frac{m \cdot A^2}{n} - A \leq (m-1) \cdot (k-1)$$

And after multiplying by n we get, if we assume $A^2 > n(k-1)$:

$$m \leq n \cdot \frac{A - (k-1)}{A^2 - n \cdot (k-1)}$$

So far we have shown that if you assume $A^2 > n(k-1)$, you get

$$m \leq n \cdot \frac{A - (k-1)}{A^2 - n \cdot (k-1)}$$

If you take $A = \sqrt{n \cdot k + 1}$, you get $m \leq \left(\sqrt{n \cdot k + 1} - (k-1)\right) \cdot n$ And so we get $m \leq n$, as wanted. □

An Algorithm for List Decoding

Our problem now is this: given a codeword $w = (w_1, \ldots, w_n)$, we are looking for polynomials who agree with it on at least $2 \cdot \sqrt{n \cdot (k-1)}$ coordinates. Intuitively, we will try to construct a polynomial in y, z so that for

Notation 3. *Let $\{f_i(y)\}_i$ denote the set of all polynomials that agree with w in at least A coordinates.*

Our polynomial will be $Q(y,z) = \prod (z - f_i(y))$. In order to get such a polynomial, we will demand that $Q(\alpha_i, w_i) = 0$ for all is.

Assuming we already have this Q, simply factorizing it as a polynomial in z, and taking every irreducible polynomial of the form $(z - g_i(y))$ in it's factorization, will give us exactly all we need.

In detail, we will define a polynomial $Q(y,z)$ in the following method: Let dz denote the degree of Q as a polynomial in z, and similarly dy. The polynomial can be written down as $\sum_{i=0}^{dz} \sum_{j=0}^{dy} a_{ij} \cdot z^i \cdot y^j$. We demand that $Q \not\equiv 0$, and

$$\forall i : \; Q(\alpha_i, w_i) = 0 \tag{8}$$

These are n constraints over $(dz+1)(dy+1)$ parameters. As long as we make sure:

$$(\deg_z(Q) + 1) \cdot \left(\deg_y(Q) + 1\right) \leq n \tag{9}$$

Then there is a solution, which gives us Q.

Claim 4. *For all i, $(z - f_i(y)) \,|\, Q(y,z)$, as polynomials in z over $F_q[y]$*

Proof. Recall that by the way it's defined in (3), f_i and w agree on at least A coordinates. We now look in the polynomial of the narrowing of Q on $(y, f_i(y))$:

$$Q_i(y) := Q(y, f_i(y))$$

Now, we know that Q_i vanishes on A of it's coordinates: for every j in which $f_i(\alpha_j) = w_j$,

$$Q_i(\alpha_j) = Q(\alpha_j, f_i(\alpha_j)) = Q(\alpha_j, w_j) \overset{(8)}{=} 0$$

Now if we can claim that the degree of Q_i is at most A, we will be done.

$$dy + (k-1) \cdot dz \leq A \tag{10}$$

is sufficient, because of the degree of the f_is.

For example, choosing

$$dy = \sqrt{n \cdot (k-1)}$$

and

$$dz = \frac{\sqrt{n \cdot (k-1)}}{k-1}$$

satisfies both (10) and (9), as wanted. □

To sum up our algorithm, we will write down the linear inequalities on $\{a_{ij}\}_{i,j=1}^{dy,dz}$ for the suggested dy,dz, and solve them, to get Q. Then we will look at the Q as a polynomial in z over $F_q[y]$, and factorize it (it's an easy problem). We are expected to get at most dz irreducible factors. For each factor of the form $(z - g(y))$, we will check and see if g agrees with w on at least A coordinates. If it does, we will return it.

Now we would like to get rid of the 2-factor. In order to do so, we will notice that the bounds we used in (10) were rough. We wanted to get

$$\deg_y(Q_i) = \deg_y\left(Q\left(y, f_i\left(y\right)\right)\right) \leq A$$

and we binded it by claiming:

$$\deg_y(Q(y, f_i(y))) \leq \deg_z(Q) \cdot \deg_y(f_i)$$

But in fact, if we look carefully, we will notice that the bound we really want is for each monomial. So we are looking for each monomial of the form $t \cdot y^i z^j$ to hold $i + (k-1) \cdot j \leq A$. How many are there: $0 \leq i \leq A$, and $0 \leq j \leq \frac{A}{k-1}$, and we want to count only the triangular part of it.

Kakeya's Problem

We are dealing with the following question: How small can a subset $\subset R^2$ be, so you could still rotate a $10cm$ needle inside it? Obviously it will contain $\aleph$ points, but it is possible to show one who's Area goes to zero.

And now to the much more natural problem- Kakeya in Finite Fields. We call $K \subseteq {F_q}^n$ a Kakeya Set if for every direction there is a starting point so that the line starting at this point and going in the direction, is all inside the set. I.E,

$$\forall y \in {F_q}^n \setminus 0 \, \exists x \in {F_q}^n \text{ so that } \{x + t \cdot y |\, t \in {F_q}^n\} \subseteq K$$

Conjecture 5. *The size of a Kakeya Set in ${F_q}^n$ is greater than $C_n \cdot q^n$, where C_n is a constant that depends on the dimention n, but not on q.*

Let $N \subset {F_q}^n$ a subset so that for each $x \notin N$, there is a line that contains x, and all of the line but x is inside N.

If we have K, a Kakeya Set, we can construct a Nikoym Set by simply taking the union over copies of K, with a different coefficient:

$$N := \cup_{t \neq 0} t \cdot K = \{t \cdot x |\, x \in K, t \neq 0\}$$

Trivially we get

$$|N| \leq q \cdot |K|$$

Theorem 6 (Lower Bound on the Size of a Nikodym Set)**.** *Let N be a Nikodym Set. So $|N| > C_n q^n$*

Proof. The idea of the proof will be finding a polynomial $f(x_1, \ldots, x_n)$ who isn't $\equiv 0$, but vanishes on every point of N. Then we will show that the polynomial must also vanish on the entire ${F_q}^n$, which will be the contradiction we need. We will do so using the fact that for every $x \notin N$, there is a line with x the only point on it who's not in N. A polynomial that vanishes on every other point in the line but x, and has a certain degree, will have to vanish on x.

Let's assume, toward contraction, $|N| < \binom{n+q-2}{n}$. We will find a polynomial f with $\deg(f) \leq q-2$ who's not the zero polynomial ($f \not\equiv 0$), and that vanishes on N ($f|_N \equiv 0|_N$). It is possible, because the number of monomials in n variables of deg $\leq q-2$ is exactly $\binom{n+q-2}{n}$, and we have $|N|$ constraints. We will now show that in fact $f \equiv 0$. Let $x \notin N$. By definition of a Nikodym Set, there exists a direction $y \in {F_q}^n$ so that $l_y := \{x + ty | \, t \in {F_q}^n \setminus 0\} \subset N$ Now we will look at f narrowed to l_y:

$$f_{x,y}(t) := f(x + ty)$$

$f_{x,y}$ is a polynomial of one variable, and it's degree: $\deg_t(f_{x,y}) \leq \deg(t) \leq q - 1$. $f_{x,y}$ was created by combining f with a linear transformation. Since we know $f|_N \equiv 0$, we get at least $q - 1$ roots for $f_{x,y}$. We recall (from number theory) that for a polynomial of degree d who's $\not\equiv 0$, can have no more than $d \cdot q^{n-1}$ roots in ${F_q}^n$. (prove by induction.)

To wrap up, we will use an estimate for Newton's Binomial:

$$\binom{q+n-2}{n} = \cdots = \frac{q^n}{n!} = C_n \cdot q^n$$

$\Rightarrow$ Every Kakeya Set K, has $|K| \geq \frac{1}{n!} \cdot q^{n-1}$ $\square$