6.440 Essential Coding Theory Mar 18, 2013

Lecture 12 DRAFT

Lecturer: Madhu Sudan *Scribe: Efrat Engel*

In this lecture we will talk about list decoding. The plan is:

- Combinatorics
- Algorithms (for Reed-Solomon codes)

1 List Decoding Combinatorics

We use list decoding of codes to be able to correct more than $\frac{d-1}{2}$ errors. A list decoding is considered good if

- It outputs a small list (poly(n))
- The list should include the transmitted message

Definition 1 *A code $C \subseteq \Sigma^n$ is (ρ, L)-list decodable if for any $y \in \Sigma^n$ we have $|\{x \in \Sigma^n | \Delta(x, y) \leq \rho n\} \cap C| \leq L$.*

In this class we will ignore the parameter L, and assume it is poly(n).

How does ρ correspond to R, δ? For any code, we have $\rho \geq \frac{\delta}{2}$ (Hamming bound). We can also prove $\rho \geq 1 - \sqrt{1-\delta}$ (Johnson bound when $|\Sigma| \to \infty$). Note that when $\delta \to 0$ this is about the same as the Hamming bound. For $|Sigma| = 2$, the Johnson bound gives $\rho \geq \frac{1}{2}(1-\sqrt{1-2\delta})$. For a large alphabet, we know δ can tend to 1, so ρ can get pretty good. We will be interested in finding $\rho \geq 1 - \sqrt{1-\delta}$.

Rate vs. list decodability: Clearly we can't give lower bound for ρ based on R for **any** code, because the rate of a code can be made arbitrarily large by adding redundancy, but we can give existential results. For any code C, we have $R \leq 1 - H_q(\rho)$, so good list decoding implies not very good rate (this is reminiscent of the Gilbert-Varshamov bound but better, and it follows from Shannon's converse by picking one solution from a given list - this will be correct with some probability). On the other hand, for any ϵ there exist codes with $R \geq 1 - H_q(\rho) - \epsilon$ (when $q \to 0$ this is about $1 - \rho - \epsilon$). The simplest proof is by a random code - exercise. For linear codes this is a more recent result, and basically random codes also work in that case.

2 List Decoding for Reed-Solomon Codes

2.1 The Basic Algorithm

We will now describe a list decoding algorithm for Reed-Solomon codes that was proposed by Madhu Sudan in the late 90's. Consider a Reed-solomon code with degree parameter $k = n^{\frac{1}{3}}$ (very redundant). Recall that regular error correcting algorithms require that half of the received information (half of $n - n^{\frac{1}{3}}$) is correct. We want to use list decoding to correct $n - n^{\frac{5}{6}}$ errors (call this $n^{\frac{5}{6}}$ agreement) to achieve the Johnson bound.

The list decoding problem for Reed-Solomon codes: For input

- $\mathbb{F}_q$, n, k (the code parameters)
- $\alpha_1, ..., \alpha_n$ (the points where the polynomial is evaluated)
- $\beta_1, ..., \beta_n$ (the received message)
- The size of the agreement a (which we will bound later)

Output all polynomials P such that $deg(P) < k$ and $|\{i|P(\alpha_i) = \beta_i\}| \geq a$.

How do we approach such a task? Consider the set of points (α_i, β_i) in $\mathbb{F}_q^2$. The Berlekamp-Welch algorithm tried to "explain" these points by claiming that all the points satisfy the equation $N(x) = yE(x)$ and then substitute them. Instead, we will try to forget about the field and the extra structure and simply find a non-zero polynomial $Q(x, y)$ such that $Q(\alpha_i, \beta_i) = 0$ for all i, hoping that this would help us find the polynomials P. Some intuition as to why this should work - Suppose there exist two polynomials $P_1(x), P_2(x)$ such that half of the points satisfy $y - P_1(x) = 0$ and the other half of the points satisfy $y - P_2(x) = 0$. Then all the points satisfy the equation $(y - P_1(x))(y - P_2(x)) = y^2 - (P_1 + P_2)(x)y + (P_1P_2)(x) = 0$.

Lemma 2 *Let $\alpha_1, ..., \alpha_n$, $\beta_1, ..., \beta_n$ be such that $(\alpha_i, \beta_i) \neq (\alpha_j, \beta_j)$ for all i. The there exists a polynomial $Q(x, y)$ with $deg_x(Q), deg_y(Q) \leq \sqrt{n}$ such that $Q(\alpha_i, \beta_i) = 0$ for all i.*

Proof Write $Q(x, y) = \sum_{j,l=0}^{\sqrt{n}} q_{jl} x^j y^l$ and solve for the indeterminates q_{jl}. There are $(\sqrt{n} + 1)^2$ indeterminates and substituting each point (α_i, β_i) in the equation $Q(x, y) = 0$ gives one homogeneous linear equation (a total of n equations), so by dimension counting there exists a non-zero solution (in fact, there are many non-zero solutions). Moreover, finding such a solution can be done efficiently, because it is simply solving a system of linear equations. ■

What do we do next? consider the intuition example above. Suppose we found $R(x), S(x)$ such that $y^2 + R(x)y + S(x) = y^2 - (P_1 + P_2)(x)y + (P_1P_2)(x) = (y - P_1(x))(y - P_2(x))$, how could we find P_1, P_2? By factoring out polynomial. Note that polynomials have unique factorization, and that factorization

of bivariate polynomials can be done efficiently. Given $Q(x,y) \in \mathbb{F}_{p^t}[x,y]$ with $deg(Q) \leq D$, it can be factored in time $poly(p,t,D)$ by a deterministic algorithm, or in time $poly(log(p),t,D)$ by a probabilistic algorithm (see the work of Lenstra, Kaltofen, Gregorier).

Lemma 3 *Suppose*

- $Q(x,y)$ *is a non-zero polynomial with* $deg_x(Q), deg_y(Q) \leq D$
- $Q(\alpha_i, \beta_i) = 0$ *for all* i
- $P(x)$ *is a polynomial with* $deg(P) < k$
- *Let* $A = \{i | P(\alpha_i) = \beta_i\}$, $|A| \geq a$

Then $(y - P(x)) | Q(x,y)$*, provided that* a *is at least some value (to be determined later).*

Proof If we wanted to prove that $(y-\eta)|Q(y)$, we could do that by verifying that $Q(\eta) = 0$. Similarly, in order to prove that $(y - P(x)|Q(x,y)$, we verify that $Q(x, P(X)) = 0$. Let $g(x) = Q(x, P(X)) = \sum_{j,l=0}^{D} q_{jl} x^j P(x)^l$, then since $deg(P) < k$, we get that $deg(g) < (k+1)D$. In order to prove that $g(x)$ is identically zero, it suffices to show that it vanishes on $(k+1)D$ points. Note that for $i \in A$ we have $g(\alpha_i) = Q(\alpha_i, P(\alpha_i)) = Q(\alpha_i, \beta_i) = 0$, hence the proof is complete provided that $|A| = a \geq (k+1)D$. ■

For $k = n^{\frac{1}{3}}$, setting $D = \sqrt{n}$ (by Lemma 1) we get the requirement $a \sim n^{\frac{5}{6}}$. We can now describe the list decoding algorithm:

1. Find non-zero $Q(x,y)$ such that $deg_x(Q), deg_y(Q) \leq \sqrt{n}$ and $Q(\alpha_i, \beta_i) = 0$ for all i.
2. Factor $Q(x,y)$ and report all polynomials $P_j(x)$ such that $(y - P_j(x)) | Q(x,y)$ and $deg(P_j) < k$.

2.2 Problems and Improvements

The construction above does not work for $k > \sqrt{n}$, because in that case it requires more agreement than the number of points. However, this can be fixed as follows: Note that in Lemma 1, we only used the degree of Q to bound the number of its coefficients. If we pick $deg_y(Q) \leq L$ and $deg_x(Q) \leq \frac{n}{L}$ for some L then Lemma 1 still works. If we write $deg_y(Q) \leq L$, $deg_x(Q) \leq D$ in Lemma 2, then we get $deg(g) < D + kL$ (and we require $a > deg(g)$). Hence if we pick $L = \sqrt{\frac{n}{k}}$ and $D = \frac{n}{L} = \sqrt{nk}$ then the construction above works and for $k = n^{\frac{1}{3}}$ we get $a = 2n^{\frac{2}{3}}$ - an improvement.

?? another improvement by picking different monomials in Q ??

The above algorithm proves that Reed-Solomon codes are $(1-\sqrt{\frac{2k}{n}}, \sqrt{\frac{2k}{n}})$-list decodable. Can we prove this combinatorially (i.e. not through an algorithm)? The answer is yes. Suppose $p_1(x), ..., p_L(x)$ are a list of solutions for the list decoding problem. Define $A_j = \{i | p_j(\alpha_i) = \beta_i\}$. Then $|A_j| \geq a$, $|A_j \cap A_l| \leq deg(p_j), deg(p_l) < k$ (two distinct polynomials cannot agree on more points than their degree) and clearly $|\bigcup_{j=1}^{l} A_j| \leq n$. We can use an inclusion-exclusion bound to get $n \geq |\bigcup_{j=1}^{l} A_j| \geq \sum_{j=1}^{l} |A_j| - \sum_{j<l} |A_j \cap A_l| \geq La - \binom{L}{2}k$, which implies an upper bound for L that comes out exactly as required.

Consider the following setting: Suppose each p_j meets each p_l exactly $k-1$ times and all the intersection points are distinct, and let the α_i's be exactly all the points of intersection. Then $n = \binom{L}{2}(k-1)$ and $a = (L-1)(k-1)$. In this case, the inclusion-exclusion bound doesn't give anything, and so does the algorithm (the algorithm doesn't find the polynomial). In order to handle this situation, we can fix the algorithm by requiring that Q has zeroes **with high multiplicity** m at the points (α_i, β_i) in step 1. With the improved algorithm, Reed-Solomon codes are $(1-\sqrt{\frac{k}{n}}, poly(n))$-list decodable.

Note that we now have more constraints on Q (3? in the above setting), so we need to require $deg_x(Q), deg_y(Q) > 3n$. Pick $deg_x(Q) < \sqrt{3nk}$, $deg_y(Q) < \sqrt{\frac{3n}{k}}$, then we get $deg(g) < 2\sqrt{3nk}$, and $g(\alpha_i)$ will now be a zero of multiplicity 2. This means that it suffices to have less zeroes α_i, so we can require $|\{i | P(\alpha) = \beta_i\}| > \sqrt{3nk}$. Thus we have achieved $((1-\sqrt{1-\delta}), poly(n))$-list decodability.

In the next lecture we will discuss folded Reed-Solomon codes, that are $(\delta - \epsilon, poly)$, or in fact $(1 - R - \epsilon, poly)$.