CS225: Pseudorandomness Prof. Salil Vadhan

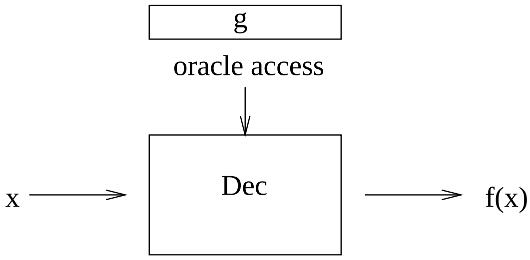
Lecture 21: Local List-Decoding

April 26, 2007

Based on scribe notes by Kevin Matulef.

1 Local List Decoding

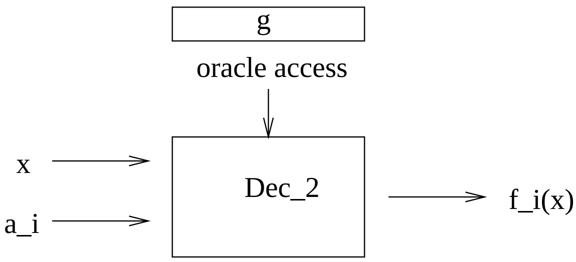
In previous lectures, we talked about a local decoding algorithm as a probabilistic algorithm which, when given oracle access to a function g close to some codeword $\hat{f}$, and given an input x, would output $\hat{f}(x)$ with high probability. Pictorially, this is shown below:



In order to decode from distances close to 1/2 with a binary code, we would like to formulate a notion of local *list*-decoding. This is slightly trickier to define, since for any function g, there may be several codewords $\hat{f}_1, \hat{f}_2, ..., \hat{f}_s$ that are close to g. So what should our decoding algorithm do? One option would be for the decoding algorithm, on input x, to output a list $\hat{f}_1(x), f_2\hat{(}x), ...\hat{f}_s(x)$. However, rather than outputing each of these values, we want to be able to specify to our decoder *which* $\hat{f}_i(x)$ to output. We do this with a two-phase decoding algorithm. The probabilistic algorithms that accomplish these phases will be referred to as Dec_1 and Dec_2:

1. Dec_1, using g as an oracle, returns a list of advice strings $a_1, a_2, ..., a_2$, which can be thought of as "labels" for each of the codewords close to g.
2. Dec_2 (again, using oracle access to g), takes input x and a_i, and outputs $\hat{f}_i(x)$.

The picture for Dec_2 is much like our old decoder, but it takes an extra input a_i corresponding to one of the outputs of Dec_1:



More formally,

Definition 1 *A* local δ list-decoding algorithm *for a code* Enc *is a pair of probabilistic oracle algorithms* $(\mathrm{Dec}_1, \mathrm{Dec}_2)$ *such that for all received words* g *and all codewords* $\hat{f} = \mathrm{Enc}(f)$ *with* $\Delta(\hat{f}, g) < \delta$*, the following holds. With probability greater at least* $1/2$ *over* $(a_1, ..., a_s) \leftarrow \mathrm{Dec}_1^g$*, there exists an* $i \in [s]$ *such that*

$$\forall x, \Pr[\mathrm{Dec}_2^g(x, a_i) = f(x)] \geq 2/3.$$

To help clarify this definition, we make the following remarks. First, we don't require that for all $j, \mathrm{Dec}_2^g(x, a_j)$ are codewords, or even that they're close to s; in other words some of the a_j's may be junk. Second, we don't explicitly require a bound on list size s, but certainly it is less than $\mathrm{time}(\mathrm{Dec}_1)$.

As we did for locally (unique-)decodable codes, we can define a *local δ list-decoding algorithm for codeword symbols*, where Dec^2 should recover arbitrary symbols of the codeword $\hat{f}$ rather than the message f. As before, this implies the above definition if the code is systematic.

Two lectures ago, we explained how having a local decoding algorithm and a worst-case hard function implied having an average-case hard function. Similarly, if we have a local list-decoding algorithm, we can make the following statement:

Proposition 2 *If* Enc *has a local δ-list decoding algorithm* $(\mathrm{Dec}_1, \mathrm{Dec}_2)$*, and* f *is worst-case hard for non-uniform time* $t = t(\ell)$*, then* $\hat{f} = \mathrm{Enc}(f)$ *is* (t', δ)*-hard, where* $t' = t/\mathrm{time}(\mathrm{Dec}_2)$.

Proof: Suppose that $\hat{f}$ is not (t', δ)-hard. Then some algorithm A running in time t' computes $\hat{f}$ with error probability smaller than δ. But if Enc has a local δ list-decoding algorithm, then (with A playing the role of g) that means there exists a_i (one of the possible outputs of Dec_1^A), such that $\mathrm{Dec}_2^A(\cdot, a_i)$ computes $f(\cdot)$ everywhere. The running time of $\mathrm{Dec}_2^A(\cdot, a_i)$ is $\mathrm{time}(A) \cdot \mathrm{time}(\mathrm{Dec}_2) = t$. Note that here we are using nonuniformity crucially to hardwire a_i as advice, in order to select the right function from the list of possible decodings. ■

2 Local List-Decoding Reed–Muller Codes

Theorem 3 *There is a universal constant* c *such that the* m*-variate Reed–Muller code of degree* d *over a finite field* $\mathbb{F}$ *can be locally* $(1-\varepsilon)$*-list decoded in time* $\mathrm{poly}(|\mathbb{F}|, m)$ *for* $\varepsilon = c\sqrt{d/|\mathbb{F}|}$.

Note that the distance at which list-decoding can be done approaches 1 as $|\mathbb{F}|/d \to \infty$. It matches the bound for Reed–Solomon codes (up to the constant c) with the benefit of sublinear-time decoding for large enough m; however, the rate is worse than for Reed–Solomon codes.

Proof: Suppose we are given an oracle $g : \mathbb{F}^m \to \mathbb{F}$ that is $(1-\varepsilon)$ close to some unkown polynomial $p : \mathbb{F}^m \to \mathbb{F}$, and that we are given an $x \in \mathbb{F}^m$. Our goal is is describe two algorithms, Dec_1 and Dec_2, where Dec_2 is able to compute $p(x)$ using a piece of Dec_1's output (i.e. advice).

The advice that we will give to Dec_2 is the value of p on a single point. Dec_1 can easily generate a (reasonably small) list that contains one such point by choosing a random $y \in \mathbb{F}^m$, and outputting all pairs (y, z), for $z \in \mathbb{F}$. In sum:

Algorithm Dec_1^g :

- choose $y \xleftarrow{\text{R}} \mathbb{F}^m$
- output $\{(y, z) \ : \ z \in \mathbb{F}\}$

Now, the task of Dec_2 is to calculate $p(x)$, given the value of p on some point y. Dec_2 does this by looking at g restricted to the line through x and y, and using the RS list-decoding algorithm to find the univariate polynomials $q_1, q_2, ..., q_t$ that are close to g. If exactly one of these polynomials q_i agrees with p on the test point y, then we can be reasonably certain that $q_i(x) = p(x)$. In sum:

Algorithm $\text{Dec}_2^g(x, (y, z))$

- Let ℓ be the line through x and y.
- Run RS $(1 - \varepsilon/2)$-list-decoder on $g|_\ell$ to get all univariate polys $q_1...q_s$ that agree with $g|_\ell$ in greater than an $\varepsilon/2$ fraction of points.
- If there exists a unique i such that $q_i(y) = z$, output $q_i(x)$.[1] Otherwise, fail.

Now that we have fully specified the algorithms, it remains to analyze them and show that they work with the desired probabilities. Observe that it suffices to compute p on at $> 11/12$ of the points x, because then we can apply the unique local decoding algorithm from last time. Therefore, to finish the proof of the theorem we must prove the following lemma

Claim 4 *Suppose that $g : \mathbb{F}^m \to \mathbb{F}$ has agreement at least ε with a polynomial p (i.e. g has distance less than $1 - \varepsilon$ from p) . For at least $1/2$ of the points $y \in \mathbb{F}^m$ the following holds for $> 11/12$ of lines ℓ going through y:*

1. $\text{agr}(g|_\ell, p|_\ell) > \varepsilon/2$.
2. *There does not exist any univariate polynomial q of degree at most d other than $p|_\ell$ such that* $\text{agr}(g|_\ell, q) \geq \varepsilon/2$ *and $q(y) = p(y)$.*

Proof of claim: It suffices to show (1) and (2) hold with probability 0.99 over random y, ℓ (then we can apply Markov's inequality to finish the job)

(1) holds by pairwise independence. If the line ℓ is chosen randomly, then points on ℓ are pairwise independent. So by using Chebychev's inequality, with the fact the expected agreement between $g|_\ell$ and $p|_\ell$ is simply the agreement between $g|_\ell$ and $p|_\ell$, which is greater than ε, we have

$$\Pr[\text{agreement} \leq \varepsilon/2] \leq \Pr[\text{deviation} > \varepsilon/2] < \frac{Var(\text{agreement})}{(\varepsilon/2)^2} < \frac{1}{|\mathbb{F}|(\varepsilon/2)^2}$$

[1] Here we are ignoring the parametrization of the line ℓ and simply viewing $g|_\ell$ as the q_i's as functions from ℓ to $\mathbb{F}$.

which can be made < 0.01 for a large enough choice of the constant c in $\varepsilon = c\sqrt{d/|\mathbb{F}|}$.

To prove (2), let $q_1, ...q_s$ all be degree $\leq d$ polynomials (not equal to $p|_\ell$), with agreement $\geq \varepsilon/2$ with $g|_\ell$. Then we have that

$$\Pr_{y \xleftarrow{\mathrm{R}} \ell} [q_i(y) = p(y)] \leq \frac{d}{|\mathbb{F}|}$$

since degree d polynomials can agree on at most d places. Then by the Johnson bound (applied to RS codes), we know $s = O(\sqrt{\mathbb{F}/d})$. Using this in the union bound, we have:

$$\Pr_y[\exists i \ : \ q_i(y) = p(y)] \leq \frac{d}{|\mathbb{F}|} \cdot s = O\left(\sqrt{\frac{d}{|\mathbb{F}|}}\right).$$

This can also be made < 0.01 for large enough choice of the constant c (since we may assume $\mathbb{F}/d > c^2$, else $\varepsilon = 1$ and the result is trivial). □

■

3 A Binary Code

We now obtain the binary locally list-decodable code we wanted by concatenating the above code with a Hadamard code.

Theorem 5 *For every $\ell \in \mathbb{N}$ and $\varepsilon > 0$, there is a code* Enc *mapping messages $f : \{0,1\}^\ell \to \{0,1\}$ to codewords $\hat{f} : \{0,1\}^{\hat{\ell}} \to \{0,1\}$ such that:*

1. $\hat{\ell} = O(\ell + \log(1/\varepsilon))$.
2. Enc *is computable in time* $2^{O(\hat{\ell})}$.
3. Enc *has a local* $(1/2 - \varepsilon)$ *list-decoding algorithm that runs in time* $\mathrm{poly}(\ell, 1/\varepsilon)$.

Proof: Given ℓ and ε, we choose a finite field $\mathbb{F}$ of characteristic 2 and of size $|\mathbb{F}| = \Theta(\mathrm{poly}(\ell, 1/\varepsilon))$ for a sufficiently large polynomial $\mathrm{poly}(\cdot)$ to be determined below.

As in the low-degree extension described last time, we let $H \subseteq \mathbb{F}$ of size $\sqrt{|\mathbb{F}|}$, $m = \lceil \ell/(\log|H|) \rceil$, view $f : H^m \to \{0,1\}$, and let $f_1 : \mathbb{F}^m \to \mathbb{F}$ be a low-degree extension of f of total degree at most $d = m \cdot |H| \leq \ell \cdot \sqrt{|\mathbb{F}|}$. Now we define $\hat{f} : \mathbb{F}^m \times \mathbb{F} \to \{0,1\}$ be obtained by encoding each symbol of f_1 in the Hadamard code.

We have seen that the outer (Reed-Muller) code is locally $(1-\varepsilon_1)$ list-decodable in time $\mathrm{poly}(m, |\mathbb{F}|)$ for

$$\varepsilon_1 = O(d/\sqrt{|\mathbb{F}|}) = O(\sqrt{\ell}/F^{1/4}) = O(\varepsilon^3).$$

The inner (Hadamard) code is $(1/2 - \varepsilon, \ell_2)$ list-decodable by brute force in time $\mathrm{poly}(|\mathbb{F}|)$, with a list size of $\ell_2 = O(1/\varepsilon^2)$. By a *local* list-decoding analogue of Problem 1 on Problem Set 5, we deduce that the concatenated code is locally δ-list-decodable in time $\mathrm{poly}(m, |\mathbb{F}|) = \mathrm{poly}(\ell, 1/\varepsilon)$ for

$$\delta = (1 - \ell_2\varepsilon_1)(1/2 - \varepsilon) = 1/2 - O(\varepsilon).$$

Changing ε by a constant factor gives the result. $\blacksquare$