6.896 Sublinear Time Algorithms November 23, 2004

Lecture 20

Lecturer: Iordanis Kerenidis *Scribe: Mihai Pătraşcu*

1 Locally-Decodable Codes

This lecture is about locally-decodable error-correcting codes. A locally-decodable code is a mapping $C : \{0,1\}^n \to \{0,1\}^m$ such that any position of a vector $x \in \{0,1\}^n$ can be recovered from $C(x)$ with a sublinear (typically constant) number of queries, even in the presence of a constant fraction of adversarial changes to $C(x)$.

More precisely, a code is (q, δ, ε)-locally decodable if it has an associated randomized algorithm A such that:

- $(\forall)x \in \{0,1\}^n, (\forall)y \in \{0,1\}^m : d(y, C(x)) \le \delta m$ and $(\forall)i \in [n]$, we have $\Pr[A(y,i) = x_i] \ge \frac{1}{2} + \varepsilon$.
- the algorithm A never read more than q positions of y.

We will be interested in constant q, δ and ε. Our definition allows A to be adaptive. However, it can be shown that without loss of generality A can be made nonadaptive. From now on, we assume the decoding algorithm is nonadaptive.

2 The Hadamard Code

Let us first prove that locally decodable codes exist. The Hadamard code is defined as follows: the i-th component of $H(x)$ is the inner product of x with i, read as a bit-vector in $\{0,1\}^n$. Note that $H : \{0,1\}^n \to \{0,1\}^{2^n}$, so the code has an exponential blow-up. However, we can achieve local decodability: the Hadamard code is a $(2, \delta, \frac{1}{2} - 2\delta)$-locally decodable code, $(\forall)0 \le \delta \le \frac{1}{4}$. The decoding algorithm is simple. Suppose we want to find x_j. We pick a random $k \in \{0,1\}^n$ and query the k-th position of the code. Also, we query position $k + e_j$, where "+" denotes vector addition in $\{0,1\}^n$ (bitwise xor), and e_j is zero everywhere except coordinate j. Each of the two queried positions are uniformly distributed (though, of course, they are not independent). So each position is corrupted with probability δ, and, by the union bound, they are both correct with probability $\ge 1 - 2\delta$. If they are both good, we can recover x_j by adding (xoring) the two values. This works because $x \cdot k + x \cdot (k + e_j) = 2x \cdot k + x \cdot e_j = x \cdot e_j = x_j$.

3 Known Bounds

A natural question is whether local decodability requires such a large penalty. For $q = 2$, Kerenidis and de Wolf [KdW] described a lower bound of $m = 2^{\Omega(n)}$, showing that an exponential blow-up is indeed necessary, and the Hadamard code is essentially optimal. As in all bounds that we discuss, the constants hidden by the asymptotic notation depend on δ and ε.

For $q > 2$, the known lower bounds are only polynomial. Katz and Trevisan [KT] showed a lower bound of $m = \Omega(n^{q/(q-1)})$, which was slightly improved in [KdW] to $m = \Omega((n/\lg n)^{1+1/(\lceil q/2 \rceil - 1)})$. The best known upper bound is $m = 2^{n^{O(\lg \lg q/(q \lg q))}}$; this comes from an equivalence with private information retrieval that we will not discuss. It remains an important open question to prove exponential lower bounds for $q > 2$.

In this lecture we will see an equivalence between locally decodable codes and smooth codes and present a lower bound based on [KT]. We will also summarize the exponential lower bound of [KdW] for $q = 2$.

4 Smooth Codes

A (q, c, ε)-smooth code is defined by the following properties:

- $(\forall)x \in \{0,1\}^n, (\forall)i \in [n]$ we have $\Pr[A(C(x), i) = x_i] \geq \frac{1}{2} + \varepsilon$. Note that we only require that the decoder work when the input is actually a codeword (we do not care about corruptions).
- $(\forall)i \in [n], (\forall)j \in [m]$ we have $\Pr[A(\cdot, i) \text{ reads } j] \leq \frac{c}{m}$. Note that, since the decoder is nonadaptive, the code input of A is irrelevant to this condition.

The second requirement says that the probes of the decoder should be roughly uniformly distributed. If we let $c = q$, we obtain an exact uniform distribution. Such a code then gives a private information retrieval scheme with q servers.

Lemma 1 *If a (q, δ, ε)-locally decodable code exists, then there exists a $(q, \frac{q}{\delta}, \varepsilon)$-smooth code.*

Proof Intuitively, since errors can appear anywhere (adversarily), the probes of the decoding algorithm should be pretty uniformly distributed; if there is a significant concentration, the algorithm can be made to fail by corrupting just a small portion of the code bits. This is made formal as follows. Define $S_i = \{j \mid \Pr[A(\cdot, i)] \text{ reads } j] \geq \frac{q}{\delta m}\}$. Since for every i the probabilities must sum to q, we have $|S_i| \leq \delta m$. Because this is within the allowable number of corruptions of the code, we may ignore the values in S_i. The decoding algorithm for the smooth code is as follows: simulate the algorithm for the locally decodable code; when it reads a position outside S_i, actually read that position; for all position in S_i, just assume you read zero. This is equivalent to running the old algorithm on an input at distance $\leq \delta m$ from the code word, so it correctly finds x_i with probability at least $\frac{1}{2} + \varepsilon$. ■

5 Matchings

Let us consider the Hadamard code for intuition. For every fixed i, the input of 2^n bits can be broken into 2^{n-1} disjoint pairs of bits (a matching on the bits). If we know the correct values for a uniformly random pair, we can reconstruct x_i. This turns out to be true in general. In a q-hypergraph over the bits of the input, we can find a matching (for each i) that covers nearly all bits. The size of the matching depends on ε, c, q. Formally:

Lemma 2 *Consider a (q, c, ε)-smooth code. Then $(\forall)i$, there exists a set of disjoint q-tuples of cardinality at least $\frac{\varepsilon m}{cq}$, such that each tuple is "ε-good for i". A q-tuple is ε-good for i if $\Pr_x[A(C(x), i) = x_i \mid A \text{ reads exactly this tuple}] \geq \frac{1}{2} + \frac{\varepsilon}{2}$. Here the probability is over the uniform distribution on x.*

We will not discuss the proof in this lecture. It appears in [Katz, Trevisan].

6 Completely Uniform Codes

Lemma 3 *A (q, c, ε)-smooth code can be converted to a $(q, q, \frac{\varepsilon^2}{2c})$-smooth code which is good on average over the random distribution on x.*

Note that the resulting code probes completely uniform positions, but is weaker because the correctness bounds only holds over a uniformly random choice of x.

Proof For the given i, consider the q-tuples from a good matching for i. Also partition the remaining positions arbitrarily into q-tuples. The decoding algorithm picks a random q-tuple and reads those bits. Because the tuples partition the code, the probed bits are clearly uniform. If the tuple is in the good matching, we run the old decoding algorithm based on these probed bits. Otherwise, we guess randomly.

The probability of correctness over a random x is $\frac{\varepsilon}{c}(\frac{1}{2}+\frac{\varepsilon}{2})+(1-\frac{\varepsilon}{c})\frac{1}{2}=\frac{1}{2}+\frac{\varepsilon^2}{2c}$. The first term comes from the probability that we choose a good tuple, and the probability of correctness based on a good tuple; the second term comes from random guessing based on a bad tuple. ■

7 Random Access Codes

A random access code only has the property that $(\forall) i : \Pr_x[A(C(x), i) = x_i] \geq \frac{1}{2} + \varepsilon$. This is among the weakest possible definitions of a code. The lower bound we obtain is likewise trivial:

Lemma 4 *For a random access code, $m = \Omega(n)$.*

Proof We have $I(x : C(x)) \leq H(C(x)) \leq m$. Also $I(x : C(x)) = H(x) - H(x \mid C(x)) \geq n - \sum_i H(x_i \mid C(x))$. Since we can recover x_i from $C(x)$ with probability $\frac{1}{2}+\varepsilon$, we have $H(x_i \mid C(x)) = H(\frac{1}{2}+\varepsilon)$. This is a constant less than 1. The chain of inequalities implies $m = \Omega(n)$. ■

8 The Lower Bound

Theorem 5 *For a (q, δ, ε)-locally decodable code, $m = \Omega(n^{q/(q-1)})$.*

Proof From the previous lemmas, it suffices to prove a lower bound for completely uniformly codes. We devise a random access code T by picking $O(m^{1-1/q})$ random bit positions of C. We claim T is a random access code, which implies $m^{1-1/q} = \Omega(n)$, so $m = \Omega(n^{q/(q-1)})$. The proof of the claim uses the birthday paradox. For every i, T includes a constant number of q-tuples from the matching for i. Since these are uniformly random, they hit the good set of q-tuples with constant probability, so we can decode with constant probability. Note that this is a randomized code (the code depends on our random choices for the bit positions). ■

9 Exponential Lower Bound for 2-Queries

An exponential lower bound for 2-query locally decodable codes was given in [KdW] using quantum arguments. At a high level, their proof can be presented in a similar to that of Theorem 5. From a locally decodable code C they create a quantum code T of size only $\log m$ that is a quantum random access code for x. One can show, using quantum information theory, that a quantum random access code must also have length $\Omega(n)$, which implies $m = 2^{\Omega(n)}$. This proof via quantum arguments remains the only known proof of this fact.

References

[KdW] Iordanis Kerenidis and Ronald de Wolf: Exponential lower bound for 2-query locally decodable codes via a quantum argument. STOC 2003, pp 106-115

[KT] Jonathan Katz and Luca Trevisan: On the efficiency of local decoding procedures for error-correcting codes. STOC 2000, pp 80-86