10/20/2003

Notes for Lecture 15

15 Lower bounds for locally decodable codes

We will show the first of several lower bounds for locally decodable codes. We will start with the most general results, which are also unfortunately quite weak. In particular, our first bounds will apply to LDCs allowing an arbitrary number of queries, but are only polynomial in the input size. Later we will see lower bounds which are exponential in the input size but only apply to the special case of two queries.

Throughout, unless otherwise noted, $\mathbf{C} : \{0,1\}^k \to \{0,1\}^n$ is a q-query LDC with a non-adaptive decoding algorithm which outputs the value of any message bit with probability at least $\frac{1}{2} + \epsilon$ on all inputs with at most δn corrupt entries.

Theorem 1 *With notation as above,*

(i) $n \geq poly(\epsilon, \delta, q) k^{q/(q-1)}$.

(ii) If $\mathbf{C}$ *sends* $\{0,1\}^k \to \Sigma^n$ *then* $n \geq poly(\epsilon, \delta, q) \left(\frac{k}{\log|\Sigma|}\right)^{q/(q-1)}$.

Later, we will see that, for two-query decoders, this is far from the right answer:

Theorem 2 *Suppose* $\mathbf{C}$ *is a* linear *code and* $q = 2$. *Then*

(i) $n \geq 2^{\Omega(\epsilon\delta k)}$ *(in fact,* $n \geq 2^{\Omega(\delta k/(1-2\epsilon))}$*, which is the best possible).*

(ii) If $\mathbf{C}$ *sends* $\{0,1\}^k \to (\{0,1\}^l)^n$*, then* $n \geq 2^{\Omega(\epsilon\delta k/2^l)}$.

The basic idea in the proof of Theorem 1 is as follows: We will show that we can restrict $\mathbf{C}$ to a set of $O(\sqrt{n})$ coordinates, and yet still recover $\Omega(k)$ bits of the message. But, information theoretically, $\mathbf{C}$ can only contain about $O(\sqrt{n}\log|\Sigma|)$ bits of information, so this quantity must be at least $\Omega(k)$ and the bound follows.

15.1 Smooth codes

A key intermediate step in the proof is to observe that the decoding algorithm for a locally decodable code must make queries which are roughly uniformly distributed over the input bits. For, if the distribution of queries is concentrated on too small a subset of indices, then an adversary could modify those entries and cause the decoding algorithm to fail with too high a probability.

To make this notion precise, we say that an algorithm A is a *c-smooth* decoder for $\mathbf{C}$ if:

(i) $\forall i, \forall x, \Pr\left(A^{\mathbf{C}(x)}(i) = x_i\right) \geq \frac{1}{2} + \epsilon$.

(ii) For every query of A, $\forall i, x, \Pr(A \text{ queries } \mathbf{C}(x)_i) \leq \frac{c}{n}$.

For example, if every query of A is uniformly distributed, then A is a 1-smooth decoding procedure.

Lemma 3 *With notation as above,* $\mathbf{C}$ *has a* $1/\delta$*-smooth decoding algorithm.*

Proof: Let A be a local decoding algorithm for $\mathbf{C}$. For any query of A, let S be the set of indices i such that the probability that A queries index i is greater than $\frac{1}{\delta n}$. Since these probabilities must sum to 1, $|S| \leq \delta n$. Now we define a new decoder A' which simulates A by running A on its input, except that whenever A requests an index in S, A' feeds A the value 0. Since we have "virtually" modified at most δn entries of its input, A, and therefore A', returns a correct decoding with probability at least $\frac{1}{2} + \epsilon$. $\square$

So we may henceforth assume that we have a c-smooth code.

15.2 Proof of Theorem 1

We will focus on the case $q = 2$. The generalization to multiple queries is straightforward.

Call a pair of indices (j_1, j_2) *ϵ-good* if

$$\Pr_{x \in \{0,1\}^k, A} (A \text{ computes } x_i \mid A \text{ queries } (j_1, j_2)) \geq \frac{1}{2} + \epsilon.$$

Our definition of local decodability requires worst-case decodability (that is, decoding must work for every x), which implies that the decoder works well for an average x. Then, by Markov's inequality, for every i,

$$\Pr_{\text{queries } (j_1,j_2) \text{ of } A} \left(\Pr_{x,A} \left(A^{\mathbf{C}(x)}(i) = x_i \mid A \text{ queries } (j_1, j_2) \right) \geq \frac{1}{2} + \frac{\epsilon}{2} \right) \geq \frac{\epsilon}{2}.$$

A natural way to view the good queries of the decoder is to associate with each index i a *recovery graph* G_i, where (j_1, j_2) is an edge of G_i exactly when the probability (over messages x) that A recovers x_i conditioned on making query (j_1, j_2) is at least $\frac{1}{2} + \frac{\epsilon}{2}$. For each edge e, let $w(e)$ denote the probability that edge e is queried. Then for each codeword entry (vertex) v, the probability that v is queried is exactly $\sum_{e \text{ adjacent to } v} w(e)$. When $\mathbf{C}$ is c-smooth, we know that this is at most $\frac{2c}{n}$ (when the code uses q queries, we have a q-regular hypergraph, and this at most $\frac{qc}{n}$). Our goal is to show that there is a large matching in such a graph. This will allow the birthday paradox to be applied to show that for some subset of $O(\sqrt{n})$ vertices, we cover at least one edge in the recovery graph. By definition of the recovery graph, when this happens we are able to probabilistically recover the corresponding bit of the message. The lower bound follows by information theory (we have used about $\sqrt{n}$ bits to recover a constant fraction of bits of the input).

In any graph, the size of a minimum vertex cover is at most twice the size of a maximum matching, so there exists a matching of size at least half the size of the minimum vertex cover. What can be the size of a vertex cover in our recovery graphs? From the above, each element in a vertex cover covers a probability mass at most $\frac{2c}{n}$. On the other hand, the total mass to be covered has weight at least $\frac{\epsilon}{2}$. So any vertex cover must have size at least $\frac{\epsilon/2}{2c/n} = \frac{\epsilon n}{4c}$. Therefore, there exists a matching of size at least $\frac{\epsilon n}{8c}$.

To simplify notation, let $\alpha = \epsilon/8c$. By the above, for each i, the recovery graph G_i contains a matching of size at least αn. We can now apply the birthday argument. Suppose we pick at random a set of $2\sqrt{\frac{n}{\alpha}}$ vertices of G_i. By Chernoff bounds, with probability $1 - o(1)$ at least a fraction $\Omega(\alpha)$ of these will intersect an edge in the matching, so we have covered $\Omega(\sqrt{\alpha n})$ endpoints of matching edges. Now sample another $\sqrt{\frac{n}{\alpha}}$ vertices. The probability that we fail to hit the other endpoint in an edge hit in the first sample is about

$$\left(1 - \frac{\sqrt{\alpha n}}{n}\right)^{\sqrt{n/\alpha}} \approx \frac{1}{e}$$

so with probability about $1 - \frac{1}{e} - o(1)$ we cover an edge.

By linearity of expectation, there must exist some subset of $O(\sqrt{\frac{n}{\alpha}}) = O(\sqrt{\frac{\epsilon n}{8c}})$ vertices such that for at least $\Omega(k)$ of the recovery graphs $\{G_i\}$, we cover at least one edge in G_i. Consider the restriction of $\mathbf{C}$ to this subset. By definition of the recovery graph, we recover $\Omega(k)$ bits of x with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$ given just these $O(\sqrt{\frac{\epsilon n}{8c}})$ bits. But these entries must contain at least $\Omega(k(1 - H_2(\frac{1}{2} + \frac{\epsilon}{2})))$ bits of information, so n must be $\Omega(k^2)$.

When $q > 2$, things work as before except that we must argue about recovery graphs with hyperedges. For hypergraphs, if we have a matching of size $\Omega(n)$ then choosing $O(n^{(q-1)/q})$ vertices is sufficient to cover some hyperedge with probability $\Omega(1)$. The more general result follows.