6.897 Algorithmic Introduction to Coding Theory October 28, 2002

Lecture 14

Lecturer: Madhu Sudan *Scribe: Ravi Sundaram*

1 Overview

- What is local decoding
- Locally decodable codes
- Local decoding of Hadamard codes

2 What is local decoding

Last time we looked at the list-decoding of Reed-Solomon codes. This time we look at the issue of local decodability and justify the study of Reed-Muller codes (even though Reed-Muller codes seem to be no better than Reed-Solomon codes it is the case that Reed-Muller codes are locally decodable whereas Reed-Solomon codes are not.)

A broad algorithmic goal in coding theory is finding efficient decoding algorithms. What does "efficient" mean? It is trivial to come up with exponential time decoding algorithms. We have looked at examples of polynomial time decoding algorithms and in a later lecture we will look at linear time decoding algorithms. Here we consider the question - how about sublinear time decoding algorithms? At first blush this seems an impossibility since it takes linear time even to read the input or to write the output. However it is possible to make the question meaningful by suitably changing the model.

The old model involves a Turing Machine (algorithm) that has read-only access to the input tape, read-write access to a work tape and write-only access to the output tape. In this model it takes $\Omega(n)$ units of time just to read the n-th bit of the input since the tape constrains the algorithm to linear access. It feels somehow unfair that reading the n-th bit is so much more expensive than reading the 1-st bit.

In the new model the algorithm has random (read-only) access to an oracle which when given a query index i returns r_i the i-th bit of the input (received word) in unit time. Similarly with respect to the output (codeword) - the algorithm on being presented with j is expected to output c_j the j-th bit of the output (or more typically the j-th *letter* of the output in the case where the alphabet has more than two elements). The model includes an adversary who gets to specify the desired output j as well as the input $r_1, r_2, \ldots, r_n$ - but is constrained to introduce no more than a specified fraction of errors on the input. The running time of the algorithm is the worst case running time over all j and $r_1, r_2, \ldots, r_n$. Since the adversary is allowed to introduce errors the model permits the algorithm to employ randomness for otherwise the adversary could trivially foil the algorithm. (Note that such sublinear algorithms have gained in popularity today with the increased focus on large data sets but they were not always so popular and in fact were studied for the first time in the context of program checking.) We now state more formally what it means for a code to be locally decodable.

Definition 1 *A code C is l-locally decodable upto error δ if there exists an algorithm A such that given inputs, index j and oracular access to received word $r \in \Sigma^n$, A queries r (adaptively, if needed) at most l times, tosses some random coins \$, runs for time at most $poly(l, \log n)$ and outputs $A^r(j)$ with the guarantee that $Prob_{\$}(A^r(j) \neq c_j) < \frac{1}{2}$ so long as there exists unique c such that $\Delta(r, c) < \delta n$*

Note that the above definition does not specify the behavior of A when there is no codeword close to the received word. If for a C there were to exist an A with the additional feature that A indicates situations where is no close codeword then C is said to be locally testable.

Note also that the analogous notion of local encodability does not make much sense since for any reasonable code ($\Omega(n)$ distance) the parity bits are forced to depend on a large ($\Omega(n)$) fraction of the input.

3 Locally decodable codes

We now revisit the issue of Reed-Solomon codes vs Reed-Muller codes. Observe that an $[n, k, d]$ Reed-Solomon code is not $k-1$-locally decodable since knowing the value of the polynomial at just $k-1$ points still allows the polynomial to take any value in the field on any other point. In fact any code C with dual $C^{\perp}$ of minimum distance $l+1$ is not l-locally decodable (prove as an exercise). Hence for local decodability we must consider those codes whose duals have small distance. Intuitively, for a code to be l-locally decodable it must be the case that the code projected onto the l coordinates must not induce all q^l possibilities for otherwise there is no information to be had from looking at the l places of the input.

Hadamard codes are a natural choice for local decodability since their duals are Hamming codes which have distance 3.

We will now argue that Reed-Muller codes too are a natural choice. Recall that Reed-Muller codes are m-variate polynomials over $\mathbb{F}_q$ of total degree $D < q$. Observe that when polynomial $f(x_1, x_2, \ldots, x_m)$ is restricted to a line $\mathbf{a} + t\mathbf{b} | t \in \mathbb{F}_q, \mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ it reduces to a degree D polynomial in one variable t and hence its value at $D+1$ points fixes its value on the entire line. This implies that the dual of the Reed-Muller code has distance (at most) $D+2$ and hence Reed-Muller codes are a natural choice for local decodability.

4 Local decoding of Hadamard codes

Proposition 1 *Hadamard codes are 2-locally decodable.*

Proof Recall that Hadamard codes are $[2^k, k, 2^{k-1}]_{\mathbb{F}_2}$-codes where input $a = (a_1, a_2, \ldots, a_k) \in \mathbb{F}_2^k$ is interpreted as the function $f_a(x) = \Sigma_{i=1}^k a_i x_i$ and encoded to $< f(x) >$ for all $x \in \mathbb{F}_2^k$.

The problem of local decoding is: given function r from $\mathbb{F}_2^k$ to $\mathbb{F}_2$, given that there exists function f such that $\Delta(r, f) \leq \delta 2^k$, and given $x \in \mathbb{F}_2^k$ output $f(x)$ with error at most $\frac{1}{2}$.

Intuitively, since the dual codes (Hamming codes) have distance 3 there must be a relationship between three bits of the input. Given that we need to compute $f(x)$, if we pick a random y then it is natural to consider $f(y)$ and the only other possibility $f(x+y)$ to complete the subspace. This motivates the following algorithm: pick uniformly random y and compute $f(x) = f(x+y) - f(y)$. Observe that $\text{Prob}_y(r(y) \neq f(y)) \leq \delta$. Since $x+y$ is also uniformly random (because y is uniformly random) therefore $\text{Prob}_y(r(x+y) \neq f(x+y)) \leq \delta$. Hence by the union bound $\text{Prob}_y(r(x+y) - r(y) \neq f(x+y) - f(y)) \leq 2\delta$. Thus if $\delta \leq \frac{1}{4}$ the algorithm outputs $f(x)$ with error at most $\frac{1}{2}$. ∎

The basic idea from the above proof that is extendable to the case of Reed-Muller codes is as follows: even though the code employs a large number of dimensions the fact that the code is a well behaved algebraic function implies that it is redundant in small subspaces or in other words it is locally decodable from a small number of dimensions. Thus the typical schema of local decodable algorithms is to use brute force to decode on the small dimension space obtained by projecting the input space so as to include the point of interest and one other random point.

Depending on Madhu details of the above schema for Reed-Muller codes may be provided in the next lecture.