6.S897 Algebra and Computation May 2, 2012

Locally Decodable Codes

Instructor: Madhu Sudan *Scribe: me*

1 Overview

- Motivations and Definitions
- Locally Decodable Codes via Multivariate Polynomials
- Locally Decodable Codes via Multiplicities
- Matching Vector Codes (Introduction and Construction)

2 Motivation

Today's lecture is on locally decodable codes. Current encoding schemes either involve storing all information in one block (which allows for good error correction, but is very slow–about the length of the block), or partitioning into smaller blocks (which allows for fast error correction, but the probabilty that an error exists goes to 1.) Locally decodable codes allow us to potentially get the best of both encodings. Locally decodable codes would allow us to reconstruct an arbitrary bit of the original message by restricting our search to only a small number of randomly chosen bits.

3 Definitions

Definition 3.1 (Locally Decodable Code)**.** Let $E : \Sigma^k \to \Sigma^n$ be an error correcting code. E should locally handle ϵ-fraction errors if there exists a decoding algorithm that on input $i \in [k]$, and given oracle access to a word y that is $\Delta((E(m), y) \leq \epsilon n$, is able to recover the ith bit of the message in at most l queries, and

$$Pr[D^y(i) = m_i] \geq \frac{2}{3}$$

This is true for every choice of y and i, and D makes at most l queries into y.

4 LDCs via Multivariate Polynomials

Let us examine some known codes and see how well they do.Take for example Reed-Solomon codes, which in some sense are almost locally decodable. For example, we have an RS code of degree 4, and we want to recover some ath symbol of the codeword. The minimum number of queries made is $k + 1$ before we can recover anything, because we have no information between coordinates. We are looking for something sublinear–we work with the idea of local "redundancies"– that is, we need the code to have a small set of coordinates that have dependencies. If this were the case, we could recover the ath symbol by looking at the codeword entries that correspond to the symbol. Given this, we will try to restrict RS in such a way that allows for this construction. A good place to start is low degree multivariate polynomials, which allows us to deal with locality constraints because we know that on the function values are restricted on a given line.

Let's look at bivariate polynomials first. We know a polynomial exists of degree d of at most $2h$, so we can fit or extend any function on to an $h \times h$ box. We define our alphabet as some field $\mathbb{F}_\epsilon$ with m variables, and given we can restrict to this box, our message space is $k = h^2$, and our encoding length $n = q^2$. The number of uncorrupted symbols in the word is at most $\frac{(\Delta_{code})}{n} \geq 1 - \frac{2h}{q}$. We also know that $h = (1-\epsilon)\frac{q}{2}$, and by doing some substitution, we see that the decoding rate $\frac{k}{n}$ goes to $\frac{1}{4}$ from below. Our locality constraint $l = \sqrt{n}$, which is the number of values we read on the line, which means we can apply RS decoding on the line.

The general setting for multivariate polynomials is as follows; our code is evaluating degree d polynomials over m variables:

- the encoding length $n = q^m$
- the message space $k = \binom{d+m}{m} \approx \frac{d^m}{m!} \approx \frac{(1-2\epsilon)^m}{m^m} n$
- distance $d = (1-2\epsilon)q$
- the field size $q > \frac{d}{1-2\epsilon}$
- locality $l \approx n^{\frac{1}{m}}$ (the locality is always the size of the field so we can decode on lines in the space).

Our rate is $\frac{k}{n} \geq \frac{1}{m!}(1-2\epsilon)^m$.

Let's look at some interesting choices for parameters. If we set $q = O(1)$, then the length $n = \exp(k^{\frac{1}{q-1}})$ and $l = q$ so we have sublinear decoding. If we let $m = \frac{\log k}{\log \log k}$ then our encoding length is polynomial in k and locality $l = poly \log k$ which is pretty good. We get better locality by setting $m = k^{\frac{1}{c}}$ then length $n = q^{k^{\frac{1}{c}}}$ $(q \approx c)$ and $l = c+1$. It was initially believed that the (roughly) best possible behavior was $l \to n$ which would let the rate go to $O(1)$, or let locality be $O(1)$ and then the encoding length would be exponential in k^ϵ. It was thought that nontrivial locality had to be traded off with a low rate. However it was shown by Kopparty, Saraf, and Yekhanin in 2010 that there is a class of codes constructed via multiplicities that allows us to achieve very good rates without sacrificing efficiency of local decoding algorithms.

5 LDCs via Multiplicities

The idea here is to construct a family of LDCs based on again evaluating multivariate polynomials as well as their partial derivatives. This allows us to obtain the good local-decodability of the previous codes and also achieve better rates (approaching 1) and distance (locality n^δ and rate $1-\delta$). So before with multivariate codes where we couldnt achieve good locality with $r > \frac{1}{2}$ we now achieve a rate of $\frac{2}{3}$ with about the same good locality for lower rates of previous codes.

Let's look at the simplest example of multiplicity codes. Let q be a prime power, and let $\delta > 0$ and the distance $d = 2(1-\delta)q$. The multiplicity code of order 2 does evaluations of degree d bivariate polynomials over $\mathbb{F}_q$ is analagous to the original bivariate construction. The coordinates are indexed by $\mathbb{F}_q^2$ and codewords are indexed by bivariate polynomials of degree at most d over the field. The alphabet size is instead $\mathbb{F}_q^3$. The codeword corresponding to the bivariate polynomial $p(x,y)$ is the vector of the ath symbol consisting of the evaluation of $p(a)$ as well as the partial derivatives p_x and p_y evaluated at a. This code has distance $\delta = \frac{1-d}{2q}$ due to the fact that two distinct polynomials of degree d can agree with multiplicity 2 on at most $\frac{d}{2q}$ fraction of points in the space. Since the alphabet size is q^3, the message length $k = \frac{\binom{d+2}{2}}{3} \approx \frac{4q^2}{6} \to \frac{2}{3}n$. Another thing to keep in mind is the possibility of "ambiguities"; if our polynomial doesn't go far enough in partial derivatives and it appears our function is 0? We get around this by using a variation of the standard Schwartz-Zippel lemma.

Lemma 5.1 (Standard Schwartz-Zippel). *If the degree of a polynomial $f \leq d$, and $Pr[f(a) = 0] > \frac{d}{q}$ then $f \equiv 0$*

Lemma 5.2 (Multiplicity Schwartz-Zippel). *If $deg(f) \leq d$ and $Pr \begin{bmatrix} p(a) \\ p_x(a) \\ p_y(a) \\ \vdots \end{bmatrix} > \frac{d}{2q}$, then $f \equiv 0$*

We now construct the parameters for degree d multivariate polynomials of S-multiplicity

- the number of derivatives is $\binom{m+s}{m}$, so
- the alphabet $\Sigma = \mathbb{F}_q^{\binom{m+s}{m}}$
- $d \to (s+1)q$
- $n = q^m; k = \frac{\binom{m+d}{m}}{\binom{m+s}{s}} \approx \frac{\frac{d^m}{m!}}{\frac{s^m}{m!}}$ so $k \approx (\frac{}{d}s)^m = (\frac{(s+1)q}{s})^m$, and so we get that $k \to q^m$

From the Multiplicity Schwartz-Zippel Lemma, we obtain

$$Pr \begin{bmatrix} p(a) \\ p_x(a) \\ p_y(a) \\ \vdots \end{bmatrix} > \frac{d}{(s+1)q}$$

so by letting m grow, and the order grow even faster we obtain the desired parameters: $l = n^\delta$ and $\frac{k}{n} \approx (1-\delta)$ without a tradeoff.

6 LDCs via Matching Vector Codes

Finally we look at another class of LDCs due to Yekhanin, Raghavendra, and Efremenko that are constructed from families of matching vectors. The decoding algorithm is again similar to the original RS algorithm, but we work over these families instead of low degree polynomials. The main result states that there exist codes with $l = 3$ and $n = exp(exp(\sqrt{(}logk))$, as opposed to $n = exp(k^(\frac{1}{2}))$ before. More generally, as $l = O(1)$ then $n = exp(exp(logk)^\epsilon)$.

The construction of these codes must satisfy some parameters:

- $m \in \mathbb{Z}^+$
- the field $\mathbb{F}_q$ has $m|(q-1)$ in order for $\mathbb{F}_q$ to have a primitive mth root.
- $S \subset \mathbb{Z}_m, 0 \notin S$
- must have an "S-nice" matrix $M \in \mathbb{Z}_m^{k \times n}$

Definition 6.1. A matrix M is "S-nice" if for for some $k \times n$ matrix partitioned into two parts, M_1(message) and M_2(check):

1. $(M_1)_{ii} = 0$
2. $(M_1)_{i,j} \in S \ \forall i \neq j$
3. M is closed under column addition.