6.440 Essential Coding Theory April 2, 2008

Lecture 15

Lecturer: Madhu Sudan *Scribe: Shyamnath Gollakota*

Note: This is a initial draft of the scribe notes. It will be updated later in the semester.

1 Overview

In these notes we will discuss Parvaresh-Vardy('05) Guruswami-Rudra('06) codes and their decoding algorithm.

In the previous lectures, we have talked about Reed Solomon codes and their list decodability. Specifically, if we take a RS code, $F_q^k \to F_q^n$, then it has a distance $1 - \frac{k}{n}$ and a list decoding radius $(1 - \sqrt{\frac{k}{n}})$. Therefore, if we have p fraction of errors then the list decoding rate of the code is $(1-p)^2$. But this is the best we can do?

In this direction, an existential argument can be given by considering a random code and look at the worst possible ball of radius pn and count the number of codewords that it contains. More specifically one can prove that,

Theorem 1 *There exist (p,poly) list decodable codes which have a rate rate,* $1 - p - \epsilon$, $\epsilon > 0$ *and a alphabet size* $q = f(\frac{1}{\epsilon})$.

Note that for Reed Solomon codes, the code has a alphabet size which tends to infinity as n tends to infinity. In contrast, here if we fix ϵ to say 0.01, q is effectively a constant. However, the above theorem is only a existential proof and it is not clear how to construct them and if there exists a polynomial time decoding and encoding algorithm these codes.

The rest of today's class is about a poly time decoding and encoding algorithm which was proposed by Guruswami and Rudra over a poly size alphabet in n.

2 Guruswami-Rudra '06 (based on Paravaresh-Vardy Result) Codes

They give an explicit family of codes of rate $1 - p - \epsilon$, and $q = n^{f(\frac{1}{\epsilon})}$ which can be encoded and decoded in polynomial time.

The beauty of their scheme is that, on the higher level, the proposed family of codes is structurally similar to RS codes. Specifically,

- Message is a polynomial p of degree less than k.

- Encoding is again the same as RS codes, i.e., $p(x_1), \cdots, p(x_n)$ but with the only change that we draw boxes each containing c of the above polynomial evaluations, as shown in the figure.

Thus, we have a code over the alphabet $\sigma = F_q^c$ and the code given by $C : (F_q^c)^{\frac{k}{c}} \rightarrow (F_q^c)^{\frac{n}{c}}$. The most prominent change is that we have a change in the alphabet size.

Another difference is that the points of evaluation of the function are picked in a more careful way. Specifically, if $\alpha \in F_q$ is a primitive element (α is a primitive element of a field if every non zero element in the field can be represented as a power of α, i.e., $F_q* = \{\alpha, \alpha^2, \cdots, \alpha^{q-1}\}$).

Now, we pick $n = q - 1$ and the code as $p(\alpha), p(\alpha^2), \cdots, p(\alpha^n)$, where as shown in the figure we chunk every c of them, where $c = f(\frac{1}{\epsilon})$. This code is called the folded Reed Solomon code.

What is surprising is that looking that the already existing RS codes in a different way has given a much more powerful family of codes.

3 How did they come up with this Code

In order to understand this it would be helpful to understand the sequence of works which led to this code.

Previous work on RS codes was that the list decoding radius was $1 - \sqrt{\frac{k}{n}}$. This bound remained undisputed for more than 5 years which prompted $(Kiayias - Yung'03)$ to use this seemingly hard problem to devise a cryptographic scheme. In particular, the following assumption is used in their scheme.

Assumption: Decoding more than $1 - \sqrt{\frac{k}{n}} + \epsilon$ errors in RS codes is hard.

In order to understand their scheme let us look at a sample application. Let us consider the canonical Key exchange protocol depicted in figure. Alice and bob share a secret key S and want to exchange secret information which cannot be determined by the Eavesdropper. The following protocol gives us the required secrecy.

- Let S be a subset of the set of integers from 1 to n, $S \subseteq [n]$.
- Alice wants to transmit the information, a polynomial p, to bob.
- She evaluates $(y_1, \cdots, y_n)$ and transmit them on the channel, where

$$y_i = \begin{cases} p(x_i) & \textbf{if } i \in S \\ random & \textbf{otherwise} \end{cases} \tag{1}$$

- Clearly B can figure out p provided $|S| \geq deg(p)$.
- One can choose the different parameters to make it hard for the eavesdropper to decode the polynomial.

Note that the above scheme is very similar, in purpose, to the one time pad. However, the only potential advantage it could have is that we could use the same secret key for more than one message, which is not possible with the one time pad.

So the question now, is if we use the same secret key twice, does the above assumption change? This leads us to a new error correcting code and a new problem.

3.1 New Error Correcting Problem

- Message: $(p_1, p_2) \rightarrow \{y_i, Z_i\}_{i=1}^n$.
- Decoding problem is as follows: Given $\{X_i, Y_i, Z_i\}_{i=1}^n$ we want (P_1, P_2) of degree k each, such that $P_1(x_i) = Y_i$ and $P_2(x_i) = Z_i$ as for many choices of i.

However, note that the above decoding problem works in a different error models. The error premise is that if there exists an error then it is *random*, i.e., (Y_i, Z_i) are uniform in F_q^2. The key question is can we do better in this model?

A couple of papers try to address this. In particular, Bleichenbacher-Kiayias-Yung show that one can poly-list decode from $1 - \frac{2k+n}{3n}$ fraction of errors. Coppersmith-Sudan show that we can poly-list decode from $1-O(\frac{k}{n})^{\frac{2}{3}}$ errors in the "random" error model. This is better than the $1 - \sqrt{\frac{k}{n}}$ bound on RS codes we had.

So what is the intuition for the improvement? Essentially, we have three variables (X_i, Y_i, Z_i). In the RS code we had just two variables (X_i, Y_i) which resulted a error tolerance of $1 - \sqrt{\frac{k}{n}}$. The larger number of variables can be exploited to increase the number of errors that can be tolerated. Specifically, the decoding could try to imitate the decoding algorithm for RS codes, i.e.,

- Step 1: Find $Q(x, y, z) \neq 0$ s.t. $Q(x_i, y_i, z_i) = 0$, $\forall i$. For the first approximation we can assume that $deg_{x,y,z} Q \leq n^{\frac{1}{3}}$.
- Step 2': "Stare at this thing.". We have a system of equations $A[V_Q] = [O \cdots 0]^T$. Unfortunately, this "staring" does not lead anywhere.
- Step 2: Instead we transform the above system of equations to its dual form, $wA = [0 \cdots 0]$, where the want $w_i = 0$ when i is an error and $w_i \neq 0$ when i is in S.

Detailed Analysis of the above decoding algorithm gives us the $1-O(\frac{k}{n})^{\frac{2}{3}}$ result.

4 Weaknesses/PV Questions

The above works has the following weaknesses.

- The random error model is not that appealing. Specifically we already have the Shannon model which deals with the random error model. So the question is can we do this in an adversarial model?
- The $O(\frac{k}{n})^{\frac{k}{n}}$ is not that appealing.
- May be we should really work under the original system of equations $AV_Q = [0 \cdots 0]^T$ instead of $wA = [0 \cdots 0]$.

Thus the question is what does Q look like in the original system of equations? The space of all solutions which Q can take can clearly be written as $Q(x, y, z) = A(x, y, z)(y - p_1(x)) + B(x, y, z)(z - p_2(x))$. this is a huge space of solutions since the polynomials A and B are random.

In addition, it becomes clear that "staring" at this system of equation actually loses information. Specifically, let us plot the zeros of $Q(x, y, z)$. In order to have a two dimensional plot, let us look at $Q(x, y, z)$ as $Q_x(y, z) \in F(x)[y, z]$. So we have a plot as shown in figure. Every point on the curve is a zero of the system and gives us a polynomial pair (p_1, p_2). But we want "the" point on this curve. But there are infinitely many points on this curve. So we effectively lose information since we have to choose from an infinitely number of points.

Thus, it looks like we have no hope! We need a poly sized list of polynomial pairs but we seem to be getting infinity many pairs of polynomial pairs. The cute trick used in the paper is to throw in another curve, $R_x(y, z) = 0$ into the picture. Let us assume that this curve is known to both the sender and the receiver and p_1 and p_2 lie on this new curve. Since the two curves are distinct and have finite degrees, there are only a finite number of points at which they can intersect. Thus, this simple trick has allowed us to reduce the number of possible points from infinity to finite!!.

So how do we implement this idea? The implementation has to answer a number of questions, namely

- Which $R_x(y, z) = 0$ should be use?
- How do we use this trick as a effective coding technique. Specifically, given p_1 can we find p_2 s.t. $R(p_1, p_2) = 0$.
- Can we bound the degree of p_2?

PV come up with a simple answer to these questions. Essentially the insight is that working with this ring is messy. So lets work with a field modulo a degree k irreducible monic polynomial, $F[x]/h(x)$. Now we have all the above figures in the modulo h(x) regime. y and z are fields and since we are working in a modulo h(x) regime, p_2 will be a degree k polynomial.

The new curve, $R_x(y, z)$, is picked as $y^D - z$ where D is a large degree picked later appropriately.

5 PV-code (Correlated RS codes)

- Given: h(x) a monic irreducible degree k polynomial in $F_q(x)$, D and $x_1, x_2, \cdots, x_n$. Pick field F_q, $(k \leq n \leq q)$
- Message: A polynomial $p_1 \in F_q(x)$ of degree less than k.
- Encoding: $p_2 = p_1^D (mod h(x))$; $P_1 \rightarrow \{p_1(x_i), p_2(x_i)\}_{i=1}^n$.
- Decoding Problem:
  - Given: $\{(x_i, y_i, z_i)\}_{i=1}^n$.
  - Find: List of all degree $<$ k polynomials p_1, s.t. $|\{i|(x_i, y_i, z_i) = (x_i, p_1(x_i), p_2(x_i)\}| \geq t$ elements, where $p_2 = p_1^d (mod(h(x))$. We need to solve the above for as small a t as possible.

5.1 Decoding algorithm

- Step 1: Find $Q(x, y, z) \neq 0$, where $h(x)$ does not divide $Q(x, y, z)$ s.t.
  - $Q(x_i, y_i, z_i) = 0, \forall i \in [n]$
  - $degree_x(Q) \leq k^{\frac{2}{3}} n^{\frac{1}{3}}$
  - $degree_y(Q) \leq \frac{n}{k}^{\frac{1}{3}}$
  - $degree_z(Q) \leq \frac{n}{k}^{\frac{1}{3}}$

  Note that the constraint that $h(x)$ does not divide $Q(x, y, z)$ is not hard to satisfy. If the resulting solution from the above equations gives us a $Q(x, y, z)$ which h(x) divides, then we can generate another $\bar{Q}(x, y, z) = \frac{Q(x,y,z)}{h(x)}$ which satisfies the above degree properties. We can continue this process until we get a Q which $h(x)$ does not divide.
- Step 2: Let $Q_x(y, z) = Q(x, y, z)(mod h(x))$. $Q_x(y, z) \in E[y, z]$ where E is a field in $F[x]|h(x)$. Substitute $y^D = z$. Now define, $P_x(y) = Q_x(y, y^D)$. Now we are looking for the points that are zeros of this polynomial. Find all $p_i \in E$ s.t $P_x(p_i) = 0$ and output all such p_i. This is clearly a poly time operation.

5.2 Analysis

We derive the final result with a sequence of simple claims.

Claim 1: Solution to Step 1 exists and can be found.

This is obvious.

Claim 2: $Q_x(y, z)$ must not be identically zero.

This is obvious from the definition, $Q_x(y,z) = Q(mod\ h(x))$. Since from construction, $h(x)$ does not divide $Q(x,y,z)$, $Q_x(y,z)$ is not identically zero.

Claim 3: $P_x(y) \neq 0$.

Note that this may not be the case if $z - y^D$ divides $Q_x(y,z)$. But is this possible? Since $degree_y(Q_x) \leq (\frac{n}{k})^{\frac{1}{3}}$, picking $D > \frac{n}{k}^{\frac{1}{3}}$ ensures that $z - y^D$ does not divide Q_x since it does not contain a y^D term in it.

Claim 4: $P_x(p_1) = 0$, if p_1 satisfies $S = \{i | p_1(x_i) = y_i\ and\ p_2(x_i) = z_i\} > 2k^{\frac{2}{3}}n^{\frac{1}{3}}$, where $p_2 = p_1^D mod\ h(x)$

So we have $Q(x_i, p_1(x_i), p_2(x_i)) = 0, \forall i \in S$. What does this tell about $g(x) = Q(x, p_1(x), p_2(x))$?

Clearly the degree of g is $\leq 3k^{\frac{2}{3}}n^{\frac{1}{3}}$. But, since $|S| > 2k^{\frac{2}{3}}n^{\frac{1}{3}}$, we can conclude that $g(x) \equiv 0$. Hence, $Q_x(p_1(x), p_2(x)) = 0 mod\ h(x)$ and as a result $S_x(p_1 mod\ h, p_2 mod\ h) = 0$ and thus $p_x(p_1) = 0$.

6 Things to ponder

- What did we achieve with this construction?
- What is the rate of the above code?
- What is the achievable ratio of errors.
- What would happen if we add more correlated polynomials $p_3, p_4, \cdots$?