**03684155: On the P vs. BPP problem.** 30/10/2016 – Lecture 1

# Local unique decoding

*Amnon Ta-Shma and Dean Doron*

## 1 Before we start

Error correcting codes. Relative distance and rate. Reed-Solomon, Reed-Muller, Hadamard.

### 1.1 Unique decoding of Reed-Solomon codes

This part closely follows Chapter 13 of the book by Guruswami, Rudra and Sudan [1]. We cite the theorem we will prove.

**Theorem 1.** *Given integer parameters $q \leq n < k$, there exists an algorithm that given as input a sequence of $n$ distinct pairs $\{(\alpha_i, y_i)\}_{i=1}^{n}$, $\alpha_i, y_i \in \mathbb{F}_q$, outputs the unique polynomial of degree at most $k-1$ satisfying $|\{i \in [n]\} : p(\alpha_i) \neq y_i| < \frac{n-k+1}{2}$. The algorithm runs in time* $\mathrm{poly}(n, \log q)$.

For the proof, look at [1, Chapter 13].

## 2 Local decoding

Suppose we use error correction to protect from errors. We start with a message $x \in \{0,1\}^k$ and we encode it to a codeword $c = C(x) \in \{0,1\}^n$ using a $[n, k, \delta]_2$ error correcting code. We know that we can recover from $\frac{\delta n}{2}$ errors in the sense that we can read a corrupted codeword (with at most $\frac{\delta n}{2}$ errors) and decode it to recover the original message $x$, and we can do that in polynomial time in $n$.

But, suppose we are only interested in the value of just one bit. I.e., we are not interested in recovering the whole string $x$, but rather just one bit $x_i$. Can we run in time polynomial in the length of $i$? That is., can we recover $x_i$ (for any $i$) in time $\mathrm{poly}(\log n)$? Such a procedure is called *local decoding* and we will define it soon. The general problem is still open today, but there are non-trivial solutions, and they are even successfully deployed in clouds and massive data sets.

**Definition 2.** *An $[n, k]_q$ code $C$ is $\delta$-locally-decodable by an algorithm $R$ if there exists an algorithm $R$ that given $i \in [k]$ and oracle access to $w \in \{0,1\}^n$ such that if $d(w, C(x)) \leq \delta n$ then $R(w, i) = C(x)_i$.*

The algorithm $R$ can also be randomized, and then we require that for every input $w$ with $d(w, C(x)) \leq \delta n$, for all $i \in [k]$, $R(w, i) = x_i$ with high probability over the internal random coins of $R$.

We usually look for $R$ runs in time polynomial in $\log k + \log q$.

# 3 Locally decoding Reed-Muller codes

We now describe a variant of Reed-Muller codes: Fix a field $\mathbb{F}_q$, a subset $H \subseteq \mathbb{F}_q$ and an integer $m$. Let $n = q^m$, $k = |H|^m$ and identify $H^m$ with $[k]$ and $\mathbb{F}_q^m$ with $[n]$.

To encode a string $x \in \mathbb{F}_q^k$ put it first on $H^m$. Namely, say $p(i) = x_i$ for every $i \in H^m = [k]$. Then, extend $p$ to be an $m$-variate polynomial $p : \mathbb{F}_q^m \to \mathbb{F}_q$ of degree at most $|H| - 1$ in each of the $m$ variables (why does such a $p$ exist? why is it unique?).

A popular setting of the parameters is $q = \text{poly}(|H|)$ in which case also $n = q^m = \text{poly}(|H|^m) = \text{poly}(k)$. also, we would like to work in the case where $q \gg |H| \cdot m$. One possible choice of parameters is $|H| = \log k$, $m = \log_{|H|} k = \frac{\log k}{\log |H|} = \frac{\log k}{\log \log k}$ and $q = \text{poly}(|H|) = \text{poly}(|H|, m) = \text{poly}(\log k)$.

## 3.1 The very little noise case

We first handle the case when the error rate $\delta$ is very small, and we do it by a simple interpolation. Roughly speaking, given a noisy word $f : \mathbb{F}_q^m \to \mathbb{F}_q$, we compute $p(i)$ by passing a random line $\ell$ through $i$ and reconstructing the function $p \circ \ell$. We expect $p \circ \ell$ and $f \circ \ell$ to have a good agreement so $p \circ \ell$ can hopefully be recovered efficiently.

**Theorem 3.** *For every $q \geq m(|H| - 1) + 2$ and $\delta \leq \frac{1}{3|H|m}$, the Reed-Muller code is $\delta$-locally decodable in time* $\text{poly}(m, |H|, \log q)$.

*Proof.* We are given oracle access to $f : \mathbb{F}_q^m \to \mathbb{F}_q$ such that there exists a degree $deg = m \cdot (|H| - 1)$ polynomial $p : \mathbb{F}_q^m \to \mathbb{F}_q$ so that $d(p, f) \leq \frac{1}{3(d+1)} n$, and a point $i \in [H]^m$, and we do the following:

1. Pick $z \in \mathbb{F}_q^m$ at random and let $\ell$ be the line $i + tz$ for $t \in \mathbb{F}_q$.
2. Let $\alpha_1, \ldots, \alpha_{deg+1}$ be distinct elements in $\mathbb{F}_q^\star$. For every $j \in [deg + 1]$, let $y_j = (f \circ \ell)(\alpha_j) = f(i + \alpha_j z)$. In other words we evaluate $f \circ \ell : \mathbb{F}_q \to \mathbb{F}_q$ on $deg + 1$ distinct points.
3. Interpolate to find a degree-$d$ univariate polynomial $h$ such that $h(\alpha_j) = y_j$ for every $j \in [deg + 1]$.
4. Output $h(0)$.

We prove:

**Lemma 4.** *The above algorithm outputs $p(i)$ with probability at least $\frac{2}{3}$.*

*Proof.* The randomness comes from the choice of $z$. We define $BAD_j$ to be the event that $y_j \neq (p \circ \ell)(\alpha_j)$. If none of these bad events occurred, we know the correct value of $p \circ \ell$ at $deg + 1$ distinct points. As $p \circ \ell$ is of degree at most $deg$, the polynomial $h$ found in the interpolation step is exactly the function $p \circ \ell$, and indeed $h(0) = (p \circ \ell)(0) = p(i)$.

We now bound the probability of the bad events. By our definition, $BAD_j$ happens if and only if $(f \circ \ell)(\alpha_j) \neq (p \circ \ell)(\alpha_j)$. Note that $\ell(\alpha_j)$ is a random point in $\mathbb{F}_q^m$. I.e., even by fixing $i$, for every $t$ we have that $i + tz$ is uniformly distributed in $\mathbb{F}_q^m$ when $z$ is uniformly distributed in $\mathbb{F}_q^m$. Hence, $\Pr_z[BAD_j] \leq \delta$ for every $j$. By the union bound, the probability that at least one bad event occurred is at most $(deg + 1)\delta \leq \frac{1}{3}$. □

The running time of the algorithm is $\text{poly}(m, deg, \log q) = \text{poly}(m, |H|, \log q)$. Note that we can amplify the success probability by a simple plurality vote. □

### 3.2 The constant noise case

We would like to improve the local error-correcting capability and to reach a constant $\delta$ (say, $\delta = \frac{1}{16}$). The reason we could tolerate only a small error is because we required the evaluated univariate polynomial to be absolutely error-free. Can we do better?

The idea is to allow errors. We will make sure that the number of errors in the restricted polynomial is likely to be small, so we can apply the unique decoding of Reed-Solomon codes. To make the sampling work better (so that the number of error is likely to be small) we will use a degree two curve instead of a line (so we will have two-wise independence instead of one-wise and Chebyshev instead of the union bound).

**Theorem 5.** *For every $q \geq 8m(|H|-1)+1$ and $\delta = \frac{1}{16}$, the Reed-Muller code is $\delta$-locally decodable in time* $\text{poly}(m, |H|, \log q)$.

*Proof.* We are given oracle access to $f : \mathbb{F}_q^m \to \mathbb{F}_q$ such that there exists a degree $deg = m \cdot (|H|-1)$ polynomial $p : \mathbb{F}_q^m \to \mathbb{F}_q$ so that $d(p, f) \leq \frac{1}{16}n$, and a point $i \in [H]^m$. We do the following:

1. Pick $z_1, z_2 \in \mathbb{F}_q^m$ at random and let $\Gamma : \mathbb{F}_q^2 \to \mathbb{F}_q^m$ be the degree two curve $\Gamma(t) = i + tz_1 + t^2 z_2$ passing at $i \in \mathbb{F}_q^m = [n]$.
2. Set $\ell = 4deg$ and let $\alpha_1, \ldots, \alpha_\ell$ be distinct elements in $\mathbb{F}_q^\star$. For every $j \in [\ell]$, let $y_j = (f \circ \Gamma)(\alpha_j)$.
3. Use Theorem 1 to find a degree-$2deg$ univariate polynomial $h : \mathbb{F}_q \to \mathbb{F}_q$ such that $h(\alpha_j) = y_j$ for at least $\frac{\ell+2deg-1}{2}$ of the points $\alpha_j$.
4. Output $h(0)$.

As in the previous unique decoding algorithm, we define $BAD_j$ to be the event that $(f \circ \Gamma)(\alpha_j) \neq (p \circ \Gamma)(\alpha_j)$. Observe that the points on $\Gamma$ are pairwise independent, where the randomness comes from the choices of $z_1$ and $z_2$. Namely, the random variables $(\Gamma(\alpha_1), \ldots, \Gamma(\alpha_\ell)) \in (\mathbb{F}_q^m)^\ell$ are uniform and pairwise independent (why?). This also implies that the random variables $BAD_1, \ldots, BAD_\ell$ are pairwise independent (why?).

Let $X = \sum_{j=1}^{\ell} BAD_j$. We know that $\mathbb{E}[X] \leq \delta\ell$ and by Chebyshev's inequality, and pairwise independence, we have that

$$\Pr[|X - \mathbb{E}(X)| \geq \delta\ell] \;\leq\; \frac{\text{Var}[X]}{\delta^2\ell^2} \;\leq\; \frac{\ell\delta(1-\delta)}{\delta^2\ell^2} \;\leq\; \frac{1}{\delta^2\ell} \;\leq\; \frac{1}{3},$$

for a large enough $\ell$ ($\frac{3}{\delta^2}$ is a constant, and if $\ell$ is not large enough, then $|H|, m$ are constants and we can hide everything in the big $O$ notation). If indeed $X < 2\delta\ell$ then $X < \frac{\ell-2d+1}{2}$ and the Reed-Solomon unique decoding succeeds and $h(0)$ is the correct output.

The running time of the algorithm is similar to the previous one, and as previously, the success probability can be amplified. □

## References

[1] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential Coding Theory*. 2015. Available at `http://www.cse.buffalo.edu/faculty/atri/courses/coding-theory/book`.