Decoding Reed-Muller codes over product sets

John Kim, Swastik Kopparty

Rutgers University

May 30, 2016

Overview

1. Error-correcting codes
   - Motivation
2. Polynomial-based codes
   - Reed-Solomon codes
   - Reed-Muller codes
3. Efficient decoding of Reed-Muller codes
   - Polynomial time decoder
   - Near-linear time decoder

Error-correcting codes

Goal: Send a message 'Don't shoot the boy.' through a noisy channel.

Problem: Don't shoot the boy. $\xrightarrow{\text{noise}}$ Don't shoot the box.

Easy fix: boy $\xrightarrow{\text{encode}}$ bbbooooyyy $\xrightarrow{\text{noise}}$ bbbooooyxy $\xrightarrow{\text{decode}}$ boy.

Problem 2: boy $\xrightarrow{\text{encode}}$ bbbooooyyy $\xrightarrow{\text{noise}}$ bbboooooxxy $\xrightarrow{\text{decode}}$ box.

- This repetition code corrects up to 1 error.
- Can correct more errors by repeating more times.
- But takes up a lot of space.

Distance

Question: What gave the repetition code its error-correcting powers?

- Codewords are far apart.
- Takes many errors to confuse one codeword with another.

Definition

The *minimum distance* D is the closest distance between two distinct codewords.

Can correct up to $D/2$ errors.

Decoding: Given a received word r such that there is a codeword c with $\Delta(r, c) < D/2$, find c.

Polynomial-based codes

Reed-Solomon code

- View characters as coefficients of a low-degree polynomial.
- Evaluate this polynomial at many locations.

Definition (Reed-Solomon code)

Let $S \subseteq \mathbb{F}_q$, $|S| = n$. Then the *Reed-Solomon code* C of polynomials of degree at most d is given by:

$$C = \{(p(x))_{x \in S} | p \in \mathbb{F}_q[X], \deg(p) \leq d\}.$$

$$\text{baa} \xrightarrow{\text{view as}} 2X^2 + X + 1 \xrightarrow{\text{encode}} (1, 4, 11, 22, 37, 56, 79, 106, 137).$$

$$\text{aaa} \xrightarrow{\text{view as}} X^2 + X + 1 \xrightarrow{\text{encode}} (1, 3, 7, 13, 21, 31, 43, 57, 73).$$

- Distinct polynomials of degree d agree in at most d points.
- Distance of this code is $n - d$.

Decoding Reed-Solomon codes

Reed-Solomon codes are central to coding theory and practice.

- Many applications in coding, complexity theory.
- Very extensively studied.

For Reed-Solomon codes, the decoding problem becomes the following basic 'noisy polynomial interpolation' problem:

Given: Received word $r : S \to \mathbb{F}_q$ such that there exists some polynomial $f(X)$ of degree at most d with $\Delta(r, f) < \frac{n-d}{2}$.

Find: $f(X)$.

- Efficient algorithms known since the 1960's (Solomon, Berlekamp, Massey, Berlekamp-Welsh).
- Even near-linear time algorithm exists.

Multivariate polynomial codes

Definition (Reed-Muller code)

Let $S \subseteq \mathbb{F}_q$, $|S| = n$. Then the *Reed-Muller code* C of m-variate polynomials of degree at most d is given by:

$$C = \{(p(x_1, \ldots, x_m))_{(x_1,\ldots,x_m)\in S^m} | p \in \mathbb{F}_q[X_1, \ldots, X_m], \deg(p) \leq d\}.$$

Lemma (Schwartz-Zippel Lemma)

Let $p(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$ be a polynomial of total degree at most d. Let $S \subseteq \mathbb{F}$ have size $|S| = n$. Then

$$\Pr_{x\in S^m}[p(x) = 0] \leq \frac{d}{n}.$$

Distance of Reed-Muller code is $n^m\left(1 - \frac{d}{n}\right)$.

Decoding bivariate Reed-Muller codes

Reed-Muller codes are also extensively studied.

- Important in many coding, complexity theory applications.
- Decoding problem is also a nice algebraic problem.

Given: Received word $r : S \times S \to \mathbb{F}_q$ such that there exists some polynomial $f(X, Y)$ of degree at most d with $\Delta(r, f) < \frac{n^2}{2}\left(1 - \frac{d}{n}\right) = n \cdot \frac{n-d}{2}$.

Find: $f(X, Y)$.

However, no known efficient algorithm in general.

- Only known for $d << n$ and algebraically special evaluation sets S (e.g. $S = \mathbb{F}_q$).
- We give an efficient algorithm for this problem.

Decoding bivariate Reed-Muller codes

Given: Received word $r : S \times S \to \mathbb{F}_q$ such that there exists some polynomial $f(X, Y)$ of degree at most d with $\Delta(r, f) < \frac{n^2}{2}\left(1 - \frac{d}{n}\right) = n \cdot \frac{n-d}{2}$.

Find: $f(X, Y)$.

Strategy:

- Write $f(X, Y) = \sum_{i=0}^{d} P_i(X) Y^{d-i}$, $\deg(P_i(X)) \leq i$.
- Find $P_0(X)$ first.
- $r(X, Y) - P_0(X)Y^d$ still close to $f(X, Y) - P_0(X)Y^d$.
- The polynomial $f(X, Y) - P_0(X)Y^d$ is simpler (has smaller Y-degree).
- Will help find the more complex linear coefficient $P_1(X)$.

Finding the coefficient of Y^d

$$x \in S \left(\text{——} \quad r(x, Y) \quad \text{——} \right) \xrightarrow{\text{decode}} g_x(Y) \xrightarrow{\text{guess}} P_0(x).$$

Try:

- For each $x \in S$, decode $r(x, Y)$ to polynomial of degree d within distance $(n - d)/2$ (hopefully $f(x, Y)$).
- Extract coefficient of Y^d to get a guess for $P_0(x)$.
- Decode guesses for $P_0(x), x \in S$ to polynomial of degree 0 within distance $n/2$.

Problem: Can distribute errors to force many wrong decodings.

Finding the coefficient of Y^d

$$x \in S \left(\text{——} \quad r(x, Y) \quad \text{——} \right) \xrightarrow{\text{decode}} g_x(Y) \xrightarrow{\text{guess}} P_0(x).$$

Problem: Can distribute errors to force many wrong decodings.

Fix: Also pass along uncertainty in our guess $P_0(x)$.

- The closer $r(x, Y)$ is to $g_x(Y)$, the more certain we are in our guess.
- $u_x = \frac{\Delta(r(x,Y),g_x(Y))}{(n-d)/2}$.
- Use a Reed-Solomon decoder that handles uncertainties (Forney - $O(n^2 \operatorname{polylog} n)$ time).

Finding the coefficient of Y^{d-i}

Assume: We have successfully found the higher order coefficients. Consider the received word:

$$r_i(X, Y) = r(X, Y) - \sum_{j=0}^{i-1} P_j(X) Y^{d-j}.$$

This is close to the polynomial $f_i(X, Y) = \sum_{j=i}^{d} P_j(X) Y^{d-j}$.

- f_i has Y-degree $d - i$.
- Reed-Solomon decoder for degree $d - i$ polynomials $r_i(x, Y)$ can handle more errors.
- Leads to more accurate guesses for $P_i(x), x \in S$.
- Necessary, as degree of $P_i(X)$ is i. (Decoder to find $P_i(X)$ handles fewer errors.)

Efficiency of Reed-Muller decoder

Question: What is the runtime of our Reed-Muller decoder?

- For bivariate codes, runtime is $O(n^3 \operatorname{polylog} n)$.
- For m-variate codes, runtime is $O(n^{m+2} \operatorname{polylog} n)$.
- Gap in runtime because decoding m-variate codes requires decoding $(m-1)$-variate codes with uncertainties.

Question: Can our algorithm be improved to near-linear time? I.e. $O(n^m \operatorname{polylog} n)$.

Improving the Reed-Muller decoder

$$x \in S \left(\text{——} \quad r_i(x, Y) \quad \text{——} \right) \xrightarrow{\text{decode}} g_{i,x}(Y) \xrightarrow{\text{guess}} P_i(x), u_{i,x}.$$

- Decoder has $d + 1$ iterations, one to find each $P_i(X)$.
- Each iteration does n Reed-Solomon decodings and one Reed-Solomon decoding with uncertainties.

Improvements:

- Reduce the number of Reed-Solomon decodings.
- Improve Forney's algorithm for Reed-Solomon decoding with uncertainties.

Reduce the number of Reed-Solomon decodings

- Each row decoded many times to successively larger radii.
- *List decode* to Johnson radius.
- List contains relevant polynomials for $\Theta(n)$ iterations provided $d = (1 - \epsilon)n, \epsilon > 0$.

Improve Reed-Solomon decoding with uncertainties

Forney's algorithm:

- Erase guess for $P_i(x)$ with probability $u_{i,x}$.
- Run Reed-Solomon decoder that handles errors (E) and erasures (F).
- Outputs correct polynomial if $E + F/2$ at most half minimum distance $D/2$.

Improvement:

- Using Chebyshev's inequality $\implies$ Decodes correctly with high probability provided the received word is slightly closer than $D/2$ to codeword.

Summary

- Reed-Muller codes around since 1950's, but efficient decoders known only for special cases.
- First polynomial time algorithm to decode Reed-Muller codes over arbitrary product sets.
- With minor assumptions, can be improved to near-linear time.

Thanks!