Recovery of exponents of polynomials of high degree

SFU

Jesse Elliott, Michael Monagan

Introduction

Polynomial interpolation is the problem of fitting a polynomial to a set of data. We represent this process with the Black Box model of polynomial interpolation.

The Black Box Model

Let $f \in \mathbb{Z}[x_1, x_2, ..., x_n]$. The goal is to interpolate f from a set of evaluations.

$$\xrightarrow{(\gamma_1,\gamma_2,...,\gamma_n)\in\mathbb{Z}^n} \blacksquare \xrightarrow{f(\gamma_1,\gamma_2,...,\gamma_n)\in\mathbb{Z}}$$

The interpolation method presented in this poster is a sparse interpolation method.

Definition Let $f \in R[x_1, x_2, ..., x_n]$, where R is a ring and $deg\, f = d$. Let t denote the number of non-zero terms of f and T_{max} denote the maximum possible terms of f. f is sparse if $t \ll T_{max} = \binom{n+d}{d}$.

We interpolate f modulo a set of primes $p_1, p_2, ..., p_s$ and use **Chinese remaindering** to recover the integer coefficients. We use a **Kronecker substitution** to reduce multivariate interpolation to a univariate interpolation.

Definition Let D be an integral domain. Let $f \in D[x_1, x_2, ..., x_n], f \neq 0$. Let $r = [r_1, r_2, ..., r_{n-1}] \in \mathbb{Z}^{n-1}, r_i > 0$. Let $K_r : D[x_1, x_2, ..., x_n] \to D[x]$ be the Kronecker substitution $K_r(f) = f(x, x^{r_1}, x^{r_1 r_2}, ..., x^{r_1 r_2 \ldots r_{n-1}})$. Let $d_i = deg_{x_i} f$ be the partial degrees of $f, 1 \leq i \leq n$. K_r is invertible if $r_i > d_i, 1 \leq i \leq n-1$.

Example Let $f(x, y, z) = 2x^4y^3z^2 + 3x^2yz + 7x^2z^3 + 5$. $K_r(f) = f(x, x^5, x^{5\cdot 4}) = 2x^{59} + 3x^{27} + 7x^{62} + 5x^0$.

After applying the Kronecker substitution the degree of the polynomial becomes exponential in n. In this poster we present a modular algorithm for recovering the exponents.

Problem

Let $f \in \mathbb{Z}[x_1 ..., x_n]$, with $K_r(f) = g(x) = a_1x^{e_1} + a_2x^{e_2} + \ldots + a_tx^{e_t}$, and $deg\, g = d$. let $E = \{e_1, e_2, ..., e_t\}$. Our goal is to interpolate E.

We first interpolate the exponents modulo a set of primes minus 1.

$$\to \begin{pmatrix} E \mod p_1 - 1 \\ E \mod p_2 - 1 \\ \vdots \\ E \mod p_s - 1 \end{pmatrix} \quad \textit{The order is unknown.}$$

Question: How do we pair up the exponents mod p-1?

We recover the exponents from their images mod $p-1$, and therefore first need to pair the images into sets that correspond to the right exponents. This is a challenge because their order is unknown.

$$\begin{pmatrix} E \mod p_1 - 1 \\ E \mod p_2 - 1 \\ \vdots \\ E \mod p_s - 1 \end{pmatrix} \xrightarrow{?} \begin{pmatrix} e_{11} & e_{12} & e_{13} & \ldots & e_{1t} \\ e_{21} & e_{22} & e_{23} & \ldots & e_{2t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{s1} & e_{s2} & e_{s3} & \ldots & e_{st} \end{pmatrix}$$

$\xrightarrow{Goal}$ $\downarrow \; \downarrow \; \downarrow \; \downarrow$

Interpolate E $\qquad$ $\mathbf{e}_1 \; \mathbf{e}_2 \; \mathbf{e}_3 \; \ldots \; \mathbf{e}_t$

Question: How do we recover the exponents?

How do we recover the exponents e_i from their images modulo $p_i - 1$? We recover the exponents by applying a generalized form of Chinese remaindering which does not require relatively prime moduli.

Method

Our method requires that we use smooth primes.

Definition A prime p is y-smooth if for every prime $q|p-1$ we have $q \leq y$.

We pick s smooth primes of the form $p = 2^kr + 1$, so that $LCM(p_1 - 1, p_2 - 1, ..., p_s - 1) > d$.

For each prime we interpolate a set $E \mod p-1$ by running the following procedure. See [4]

Procedure **Interpolate Exponents mod p-1:**
INPUT: Black Box polynomial g and a prime p.
OUTPUT: E mod $p-1$.
(1) Pick a generator $\alpha \in \mathbb{Z}_p$.
For $1 \leq j \leq 2t-1$ compute $v_j = g(\alpha^j)$.
(2) Compute $\lambda(z) = \prod_{i=1}^{t}(z - \alpha^{e_i \mod p-1})$ from the evaluations v_j with the Berlekamp Massey algorithm.
(3) Compute the roots of $\lambda(z) : \alpha^{e_1}, \alpha^{e_2}, ..., \alpha^{e_t}(mod\, p)$.
(4) Solve mod p-1 by taking the discrete logarithm:
$\log_\alpha(\alpha^{e_i})\, mod\, p = e_i\, mod\, p-1$. [1]

$$(Exponents \mod p-1) \to \begin{pmatrix} E \mod p_1 - 1 \\ E \mod p_2 - 1 \\ \vdots \\ E \mod p_s - 1 \end{pmatrix}$$

Sorting the Exponents mod p-1

$$p = \delta x + 1$$

The moduli we choose share a common divisor δ. That is, for all $1 \leq i \leq s, p_i = \delta x_i + 1$. If we make δ large enough then the birthday paradox tells us

$\delta > t^2 \Rightarrow Prob\{unique\, exponents\} > .6.$

If the exponents are unique mod δ then their images mod δx are unique mod δ, and therefore we can sort them mod δ.

$$\begin{pmatrix} E \mod p_1 - 1 \\ E \mod p_2 - 1 \\ \vdots \\ E \mod p_s - 1 \end{pmatrix} \xrightarrow{Sort\ mod\ \delta} \begin{pmatrix} e_{11} & e_{12} & e_{13} & \ldots & e_{1t} \\ e_{21} & e_{22} & e_{23} & \ldots & e_{2t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{s1} & e_{s2} & e_{s3} & \ldots & e_{st} \end{pmatrix}$$

Recovering the Exponents

Once we have sorted the images, we can apply a generalized form of the Chinese remainder theorem to recover the exponents.

Generalized Chinese Remainder Theorem Let $m_1, ..., m_s$ be positive integers. Let $M = LCM(m_1, ..., m_s)$, and let $u_1, ..., u_s$ be any integers. There exists exactly one integer $\mathbf{u}$ such that,
i) $\mathbf{u} \equiv u_i(mod\ m_i), 1 \leq i \leq s$, and
ii) $0 \leq \mathbf{u} \leq M$,
iff $u_i \equiv u_j\ (mod\ gcd(m_i, m_j)), 1 \leq i < j \leq s$. [2]

The theorem bellow encapsulates that if the exponents are unique mod δ, then their modular images are unique mod δ and pair up into sets from which we can recover the exponents.

Theorem Let $m_1 = \delta x_1, m_2 = \delta x_2, ..., m_s = \delta x_s$, and $M = LCM(m_1, ..., m_s)$. Let E be a set of t non negative integers. Let $E_1, ..., E_s$ be sets such that, $E_k = \{x \bmod m_k : x \in E\}, 1 \leq k \leq s$. If E contains distinct elements modulo δ, then there exists exactly t integers, $u_1, u_2, ..., u_t$ such that
(i) $0 \leq u_i < M$,
(ii) for each u_i, there exist exactly s integers $u_{1i}, u_{2i}, ..., u_{si}$ such that $u_{1i} \in E_1, u_{2i} \in E_2, ..., u_{si} \in E_s$, and $u_i \equiv u_{ki}(mod\ m_k), 1 \leq {}^\forall k \leq s$.

Example:

$E_1 = \{65, 97, 124, 189, 300\}$ $\qquad$ $p_1 = 10 \cdot 31 + 1$
$E_2 = \{377, 394, 509, 680, 965\}$ $\qquad$ $p_2 = 10 \cdot 97 + 1$
$E_3 = \{20, 34, 57, 89, 115\}$ $\qquad$ $p_3 = 10 \cdot 13 + 1$
$E_4 = \{35, 100, 119, 217, 254\}$ $\qquad$ $p_4 = 10 \cdot 33 + 1$

Notice $E_1\ mod\ \delta$ gives $E_1 = \{65, 97, 124, 189, 300\}\ mod\ 10 = \{5, 7, 4, 9, 0\}$, and therefore the exponents are unique mod $\delta = 10$. Now the images can be sorted mod δ.

$$\xrightarrow[(mod\ \delta)]{Sort} \begin{pmatrix} 300 & 124 & 65 & 97 & 189 \\ 680 & 394 & 965 & 377 & 509 \\ 20 & 34 & 115 & 57 & 89 \\ 100 & 254 & 35 & 217 & 119 \end{pmatrix} \begin{matrix} = E \mod 10 \cdot 31 \\ = E \mod 10 \cdot 97 \\ = E \mod 10 \cdot 13 \\ = E \mod 10 \cdot 33 \end{matrix}$$

Now apply the Generalized Chinese remainder theorem.

$\downarrow \quad \downarrow \quad \searrow \quad \searrow \quad \searrow$

4126090 10826564 1918655 264217 7269689

E = {264217, 1918655, 4126090, 7269689, 10826564}

g(x) = $a_1x^{264217} + a_2x^{1918655} + a_3x^{4126090} + a_4x^{7269689} + a_5x^{10826564}$

Smooth Primes

In step 4 of procedure InterpolateExponents we solve a discrete logarithm.

Question: How do we solve the discrete logarithm efficiently?

If $p = p_1^{f_1}p_2^{f_2} \ldots p_k^{f_k} + 1$, then the cost of running the Pohlig-Hellman algorithm is $O(\sum_{n=1}^{k} f_i(\log p + \sqrt{p_i}))$ [1]. Therefore to maintain an efficient algorithm we need that p is smooth.

Our application uses 63 bit primes. We need that there are enough smooth primes less than 2^{63}.

Definition An integer x is y-smooth if for every prime $p|x$ we have $p \leq y$. The number of y-smooth primes is

$$\pi(x, y) = \sum_{\text{primes } p \leq x \text{ such that } p-1 \text{ is } y-\text{smooth}} 1.$$

Example We computed $\pi(2^{30}, 1024) = 4816780$.

Theorem (Friedlander J.B., 1989 [3]). If $\alpha > \sqrt{e}/2 = 0.303...$ and $y > x^\alpha$ then there exists $c > 0$ such that

$$\pi(x, y) > c\frac{x}{\log x}$$

We know by the prime number theorem that $\pi(x) \sim \frac{x}{\log x}$. Therefore, the above theorem tells us that, for every $\alpha > .303$, we can find a constant such that $\pi(x, y) > c\pi(x)$. We computed a few of these constants which are listed in the following table.

	$\alpha = 0.5$		$\alpha = 0.3\overline{3}$		$\alpha = 0.25$	
y	x	c	x	c	x	c
2^{16}	2^{32}	0.33746	2^{48}	0.05600	2^{64}	0.00591
2^{18}	2^{36}	0.33272	2^{54}	0.05578	2^{72}	0.00558
2^{20}	2^{40}	0.33081	2^{60}	0.05418	2^{80}	0.00568
2^{22}	2^{44}	0.32957	2^{66}	0.05355	2^{88}	0.00594
2^{24}	2^{48}	0.32604	2^{72}	0.05307	2^{96}	0.00563
2^{26}	2^{52}	0.32665	2^{78}	0.05297	2^{104}	0.00545
2^{28}	2^{56}	0.32400	2^{84}	0.05171	2^{112}	0.00568
2^{30}	2^{60}	0.31983	2^{90}	0.05195	2^{120}	0.00529

References

[1] S. Pohlig and M. Hellman (1978), "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance."

[2] D. Knuth (1981), "The Art of Computer Programming - Volume 2: Semi-Numerical Algorithms", 2nd ed., 1981.

[3] J. B. Friedlander, "Shifted Primes without Large Prime Factors", pp. 393–401 in Number Theory and Applications (Banff, AB, 1988)

[4] J. Hu and M. Monagan (2016), "A Fast Parallel Sparse Polynomial GCD Algorithm", Proc. ISSAC 2016, pp. 271–278, ACM Press, 2016.

Maplesoft
Mathematics • Modeling • Simulation

cecm

NSERC CRSNG