Privacy-Preserving Distributed Linear Regression on High-Dimensional Data

Borja Balle
Amazon Research Cambridge
(work done at Lancaster University)

Based on joint work with **Adria Gascon, Phillipp Schoppmann, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans**

Motivation

Treatment Outcome	Medical Data			Census Data			Financial Data		
	Attr. 1	*Attr. 2*	…	*Attr. 4*	*Attr. 5*	…	*Attr. 7*	*Attr. 8*	…
-1.0	0	54.3	…	North	34	…	5	1	…
1.5	1	0.6	…	South	12	…	10	0	…
-0.3	1	16.0	…	East	56	…	2	0	…
0.7	0	35.0	…	Centre	67	…	15	1	…
3.1	1	20.2	…	West	29	…	7	1	…

Note: This is vertically-partitioned data; similar problems with horizontally-partitioned

PMPML: Private Multi-Party Machine Learning

Problem

- Two or more parties want to ***jointly learn*** a model of their data
- But they ***can't share*** their private data with other parties

Assumptions

- Parameters of the model will be received by all parties
- Parties can engage in on-line secure communications
- External parties might be used to outsource computation or initialize cryptographic primitives

The Trusted Party "Solution"

Receives plain-text data, runs algorithm, returns result to parties

The Trusted Party assumption:

- Introduces a single point of failure (with disastrous consequences)
- Relies on weak incentives (especially when private data is valuable)
- Requires agreement between all data providers

=> Useful but unrealistic. Maybe can be simulated?

Secure Multi-Party Computation (MPC)

Public: $f(x_1, x_2, \ldots, x_p) = y$

Private: (party i) x_i

Goal: *Compute f in a way that each party learns y (and nothing else!)*

Tools: Oblivious Transfers (OT), Garbled Circuits (GC), Homomorphic Encryption (HE), etc

Guarantees: Honest but curious adversaries, malicious adversaries, computationally bounded adversaries, coalitions

In This Talk

A PMPML system for vertically partitioned linear regression

Features:

- Scalable to millions of records and hundreds of dimensions
- Formal privacy guarantees
- Open source implementation

Tools:

- Combine standard MPC constructions (GC, OT, TI, …)
- Efficient private inner product protocols
- Conjugate gradient descent robust to fixed-point encodings

FAQ: Why is PMPML…

Exciting?

Can provide access to previously ”locked” data

Hard?

Privacy is tricky to formalize, hard to implement, and inherently interdisciplinary

Worth?

Better models while avoiding legal risks and bad PR

Related Work

Ref	Crypto	Linear Solver	Examples	Features	Running Time	Accuracy
[1]	HE	Newton	50K	22	2d	YES
[2]	HE+GC	Cholesky	2K	20	6m	YES
[3]	TI/HE	Newton	50K	223	“7h”	NO
[4]	SS	Gauss/Chol/CGD	10K	10	11s	NO

[1] Hall et al. (2011). Secure multiple linear regression based on homomorphic encryption. *Journal of Official Statistics*.

[2] Nikolaenko et al. (2013). Privacy-preserving ridge regression on hundreds of millions of records. In *Security and Privacy (SP)*.

[3] Cock et al. (2015). Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Workshop on Artificial Intelligence and Security*.

[4] Bogdanov et al. (2016). Rmind: a tool for cryptographically secure statistical analysis. *IEEE Transactions on Dependable and Secure Computing*.

Functionality: Multi-Party Ridge Regression

Training Data

$$X = [X_1 \;\; X_2] \in \mathbb{R}^{n \times d}$$

$$Y \in \mathbb{R}^n$$

Private Inputs

Party 1: X_1, Y

Party 2: X_2

Ridge Regression

$$\min_{\theta \in \mathbb{R}^d} \|Y - X\theta\|^2 + \lambda\|\theta\|^2$$

(optimization)

$$(X^\top X + \lambda I)\theta = X^\top Y$$

(closed-form solution)

Aggregation and Solving Phases

Aggregation

$$A = X^\top X + \lambda I$$

$$b = X^\top Y$$

$$\mathcal{O}(nd^2)$$

$$X^\top X = \begin{bmatrix} X_1^\top X_1 & \boxed{X_1^\top X_2} \\ \boxed{X_2^\top X_1} & X_2^\top X_2 \end{bmatrix}$$

(cross-party products)

Solving

$$\theta = A^{-1} b$$

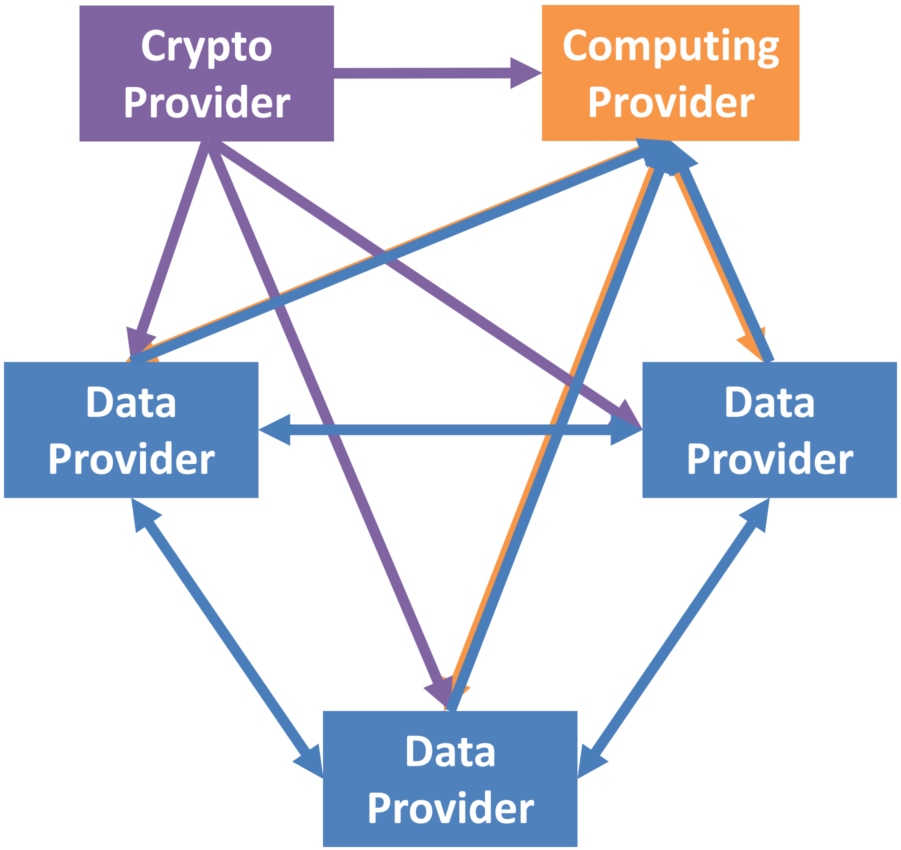
$\mathcal{O}(d^3)$ (eg. Cholesky)

Approximate iterative solver $\mathcal{O}(kd^2)$ (eg. k-CGD)

Challenges and Trade-offs

- MPC protocols: out of the box *vs.* tailored
- Encoding real numbers: speed *vs*. accuracy
- Scalability: n, d, # parties
- Privacy guarantees: semi-honest *vs*. malicious
- External parties: speed *vs*. privacy
- Interaction: off-line *vs*. on-line

Protocol Overview



Alternative: CrP and CoP simulated by non-colluding parties

Aggregation Phase

1. CrP distributes correlated randomness
2. DPs run multiple inner product protocols to get additive share of *(A,b)*

Solving Phase

3. CoP get GC for solving linear system from CrP
4. DPs send garbled shares of *(A,b)* to CoP
5. CoP executes GC and returns solution to DPs

Aggregation Phase – Two Protocols

$X_1^\top X_2$ $\longrightarrow$ $f(x_1, x_2) = \langle x_1, x_2 \rangle$

(matrix product) (inner product b/w columns)

- External pre-processing: inner product protocol leveraging correlated randomness supplied by Trusted Initializer (TI)
- Stand-alone: 2-party inner product protocol based on Oblivious Transfers (OT)

Fixed-point Encoding

$\mathcal{O}(\log(n/\varepsilon))$ bits $\Rightarrow$ error $\leq \varepsilon$

Aggregation Phase - Experiments

Trade-offs

- **OT**: stand-alone, out-of-the-box MPC
- **TI**: pre-processing, external party, faster

n	d	Number of parties: 2		3		5	
		OT	TI	OT	TI	OT	TI
$5 \cdot 10^4$	20	1m50s	1s	1m32s	2s	1m7s	2s
$5 \cdot 10^4$	100	42m12s	25s	34m39s	32s	24m58s	37s
$5 \cdot 10^5$	20	18m18s	15s	14m29s	18s	12m10s	21s
$5 \cdot 10^5$	100	7h3m56s	4m47s	5h20m52s	6m1s	4h17m8s	6m58s
$1 \cdot 10^6$	100	-	10m1s	-	12m42s	-	14m48s
$1 \cdot 10^6$	200	-	39m16s	-	49m56s	-	59m22s

* AWS C4 instances, 1Gbps

Solving Phase – Garbled Circuits

$$A\theta = b$$

(PSD linear system)

$$(A_i, b_i)$$

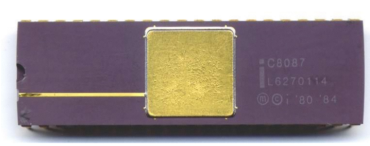
(party i's input)

$$A = \sum_i A_i \quad b = \sum_i b_i$$

Solver implemented in a Garbled Circuit

Floating-point computation with GC is not feasible (yet)

GC $\ll$



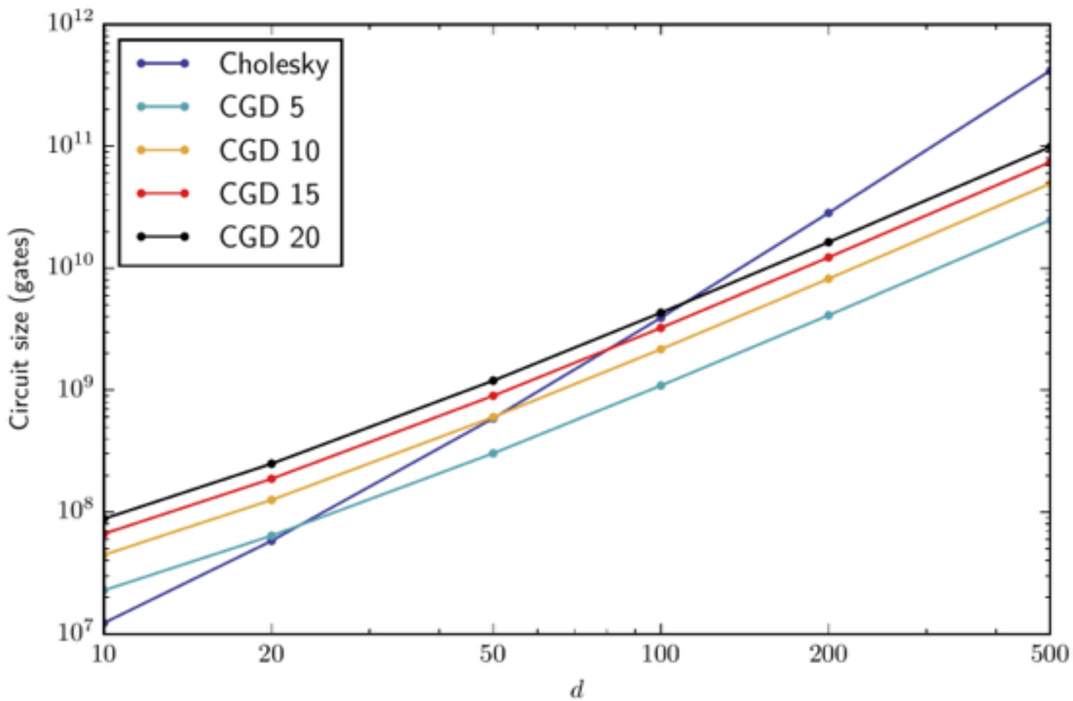
Intel 8087 FPU (1980)

Year	Device / Paper	32 bit floating point multiplication (ms)
1961	IBM 1620E	**17.7**
1980	Intel 8086 CPU (software)	**1.6**
1980	Intel 8087 FPU	**0.019**
2015	Pullonen et al. @ FC&DS	**38.2**
2015	Demmler et al. @ CCS	**9.2**

MPC

Solving Phase – Two Methods

- **Cholesky**: exact, cubic, used in [Nikolaenko et al.'13]
- **Conjugate Gradient Decent (CGD)**: approximated, "quadratic"



Fixed-point + Conjugate Gradient Descent

Textbook CGD

$$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$$

$$\mathbf{p}_0 := \mathbf{g}_0$$

repeat for $k = 1 \ldots K$

$$\alpha_k := \frac{\mathbf{g}_k^\top \mathbf{p}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

$$\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_k \mathbf{p}_k$$

$$\mathbf{g}_{k+1} := \mathbf{g}_k - \alpha_k \mathbf{A}\mathbf{p}_k$$

$$\beta_k := \frac{\mathbf{p}_k^\top \mathbf{A}\mathbf{g}_{k+1}}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

$$\mathbf{p}_{k+1} := \mathbf{g}_{k+1} - \beta_k \mathbf{p}_k$$

Fixed-point CGD

$$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$$

$$\mathbf{p}_0 := \mathbf{g}_0 / \|\mathbf{g}_0\|_\infty$$

repeat for $k = 1 \ldots K$

$$\alpha_k := \frac{\mathbf{g}_k^\top \mathbf{p}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

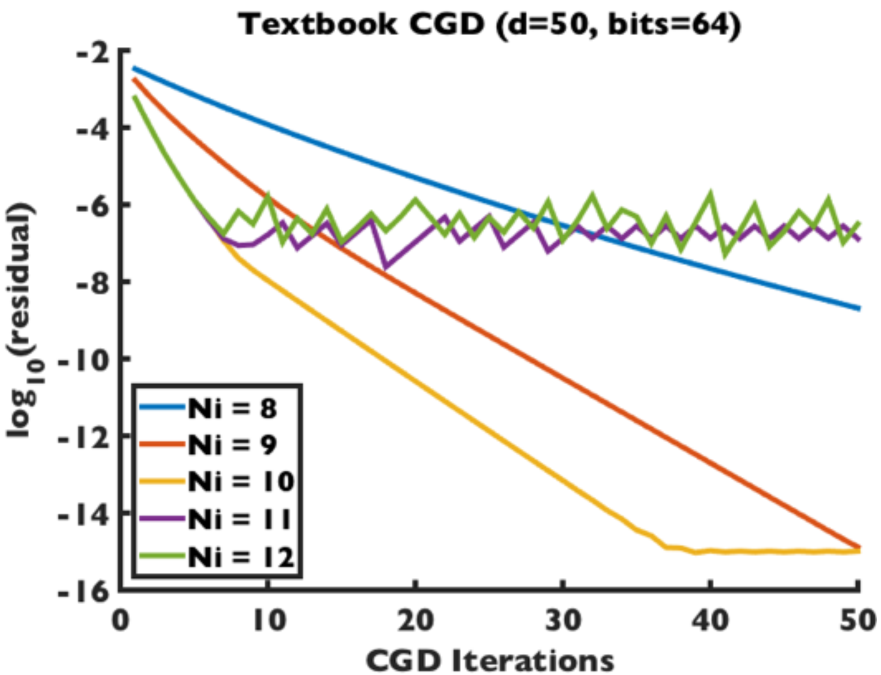
$$\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_k \mathbf{p}_k$$

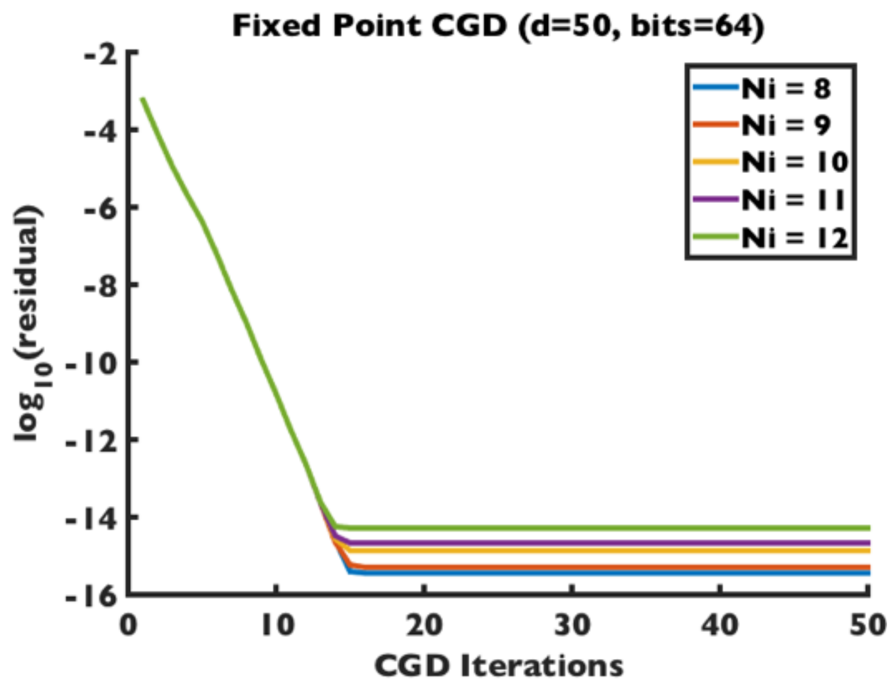
$$\mathbf{g}_{k+1} := \mathbf{g}_k - \alpha_k \mathbf{A}\mathbf{p}_k$$

$$\beta_k := \frac{\mathbf{p}_k^\top \mathbf{A}\left(\mathbf{g}_{k+1} / \|\mathbf{g}_{k+1}\|_\infty\right)}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$$

$$\mathbf{p}_{k+1} := \mathbf{g}_{k+1} / \|\mathbf{g}_{k+1}\|_\infty - \beta_k \mathbf{p}_k$$

Fixed-point + Conjugate Gradient Descent





Bits = Ni + Nf + 1

Ni = number of integer bits

Nf = number of fractional bits

Experiments with UCI Datasets

id	Name	Reference	d	n
1	Student Performance	[11, 14]	30	395
2	Auto MPG	[72]	7	398
3	Communities and Crime	[61, 62]	122	1994
4	Wine Quality	[12, 13]	11	4898
5	Bike Sharing Dataset	[23, 24]	12	17 379
6	Blog Feedback	[8, 9]	280	52 397
7	CT slices	[33]	384	53 500
8	Year Prediction MSD	[5]	90	515 345
9	Gas sensor array	[26, 27]	16	4 208 261

- 70-30 train-test random split
- Regularization tuned in the clear
- Implemented in Obliv-C
- 2+2 parties, 20 CGD iterations
- Data standardization inside protocol

- CGD faster for d > 100
- 32 bits provide good accuracy

id	Optimal	FP-CGD (32 bits)		Cholesky (32 bits)		FP-CGD (64 bits)		Cholesky (64 bits)	
	RMSE	time	RMSE	time	RMSE	time	RMSE	time	RMSE
1	4.65	19s	4.65 (-0.0%)	5s	4.65 (-0.0%)	1m53s	4.65 (-0.0%)	35s	4.65 (-0.0%)
2	3.45	2s	3.45 (-0.0%)	0s	3.45 (-0.0%)	13s	3.45 (0.0%)	1s	3.45 (0.0%)
3	0.14	4m27s	0.14 (0.3%)	4m35s	0.14 (-0.0%)	24m24s	0.14 (0.2%)	26m31s	0.14 (-0.0%)
4	0.76	3s	0.76 (-0.0%)	0s	0.80 (4.2%)	23s	0.76 (-0.0%)	4s	0.76 (-0.0%)
5	145.06	4s	145.07 (0.0%)	1s	145.07 (0.0%)	26s	145.06 (0.0%)	4s	145.06 (0.0%)
6	31.89	24m5s	31.90 (0.0%)	53m24s	32.19 (0.9%)	2h3m39s	31.90 (0.0%)	4h40m23s	31.89 (-0.0%)
7	8.31	44m46s	8.34 (0.4%)	2h13m31s	8.87 (6.7%)	3h51m51s	8.32 (0.1%)	11h49m40s	8.31 (-0.0%)
8	9.56	4m16s	9.56 (0.0%)	3m50s	9.56 (0.0%)	16m43s	9.56 (0.0%)	13m28s	9.56 (0.0%)
9	90.33	48s	95.05 (5.2%)	42s	95.06 (5.2%)	1m41s	90.35 (0.0%)	1m9s	90.35 (0.0%)

Conclusion

Summary

- Full system is accurate and fast, available as open source
- Scalability requires hybrid MPC protocols and non-trivial engineering
- Robust fixed-point CGD inside GC has many other applications

Extensions

- Security against malicious adversaries
- Classification with quadratic loss
- Kernel ridge regression
- Differential privacy at the output

Future Work

- Models without a closed-form solution (eg. logistic regression, DNN)
- Library of re-usable ML components, complete data science pipeline

Read It, Use It

http://eprint.iacr.org/2016/892

Privacy-Preserving Distributed Linear Regression on High-Dimensional Data

PETS'17 ACCEPTED

Adrià Gascón[1], Phillipp Schoppmann[2], Borja Balle[3], Mariana Raykova[4], Jack Doerner[5], Samee Zahur[6], and David Evans[7]

[1] University of Edinburgh
[2] Humboldt University of Berlin
[3] Lancaster University
[4] Yale University
[5] Northeastern University
[6] Google
[7] University of Virginia

https://github.com/schoppmp/linreg-mpc

schoppmp / **linreg-mpc**

Unwatch 6 | Star 1 | Fork 1

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs

A Secure Multiparty Computation (MPC) protocol for computing linear regression on vertically distributed datasets

388 commits | **3** branches | **0** releases | **4** contributors | GPL-3.0