Secure Linear Regression on Vertically Partitioned Datasets

Adria Gascon Phillipp Schoppmann Borja Balle
Mariana Raykova Samee Zahur Jack Doerner
David Evans

Predictive Model

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhythmia	…	Inflammation	Dysphagia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Given samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
  - $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- Learn a function f such that $f(x_i) = y_i$

Linear Regression

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhythmia	…	Inflammation	Dysphagia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Given samples $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
  - $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- Learn a function f such that $f(x_i) = y_i$

f is well approximated by a linear map
$y_i \approx \theta^T x_i$

Secure Computation

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Shared database - $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ do not belong to the same party
- Compute θ securely ($y_i \approx \theta^T x_i$)

Horizontally Partitioned Database

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Different rows belong to different parties
  - E.g., each patient has their own information

Vertically Partitioned Database

Patient	Blood Count			Heart Conditions			Digestive Track			…	Medicine Effectiveness
	RBC	WBC	…	Murmur	Arrhyt hmia	…	Inflamm ation	Dyspha gia	…		
A	3.9	10.0		0	0		0	1			1
B	5.0	4.5		1	0		1	2			1.5
C	2.5	11		0	1		1	0			2
D	4.3	5.3		2	1		0	1			1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Different columns belong to different parties
  - E.g., different specialized hospitals have different parts of the information for all patients

Ridge Regression

- Computing linear model on inputs $(x_1, y_1),\ldots, (x_n, y_n)$
  - $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
- Optimization formulation

$$\min_{\theta} \frac{1}{n}\sum_{i=1}^{n}(y^i - \langle\theta, x^i\rangle)^2 + \lambda\|\theta\|^2$$

Loss **Regularization**

- Linear System Formulation

$$X = \begin{bmatrix} \cdots & x^1 & \cdots \\ & \vdots & \\ \cdots & x^n & \cdots \end{bmatrix} \qquad Y = \begin{bmatrix} y^1 \\ \vdots \\ y^n \end{bmatrix}$$

$$\left(\frac{1}{n}X^\top X + \lambda I\right)\theta = X^\top Y$$

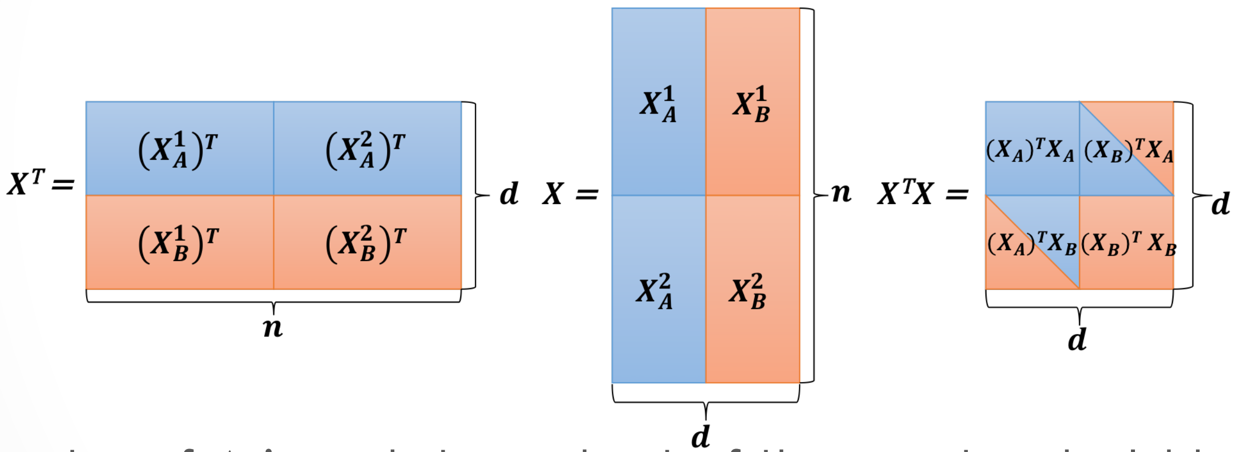
Positive definite

Contributions

- Secure computation for ridge regression for vertically partitioned database
  - Two phase protocol:
    - **Phase1** – compute $A = \frac{1}{n} X^T X + \boldsymbol{\lambda} I \quad b = X^T Y$
      - Output is additively shared between two parties
    - **Phase2** – solve $A\theta = b$ where A and b are shared between two parties
- Two party and multiparty protocol for Phase1
  - **Two party inner product computation**
- Three algorithms for Phase2:
  - **Cholesky, LDLT, Conjugate Gradient Descent (CGD)**
- Implementation and evaluation

Phase 1

- Compute $A = \frac{1}{n}X^T X + \boldsymbol{\lambda} I \quad b = X^T Y$
  - The output is additively shared between two parties

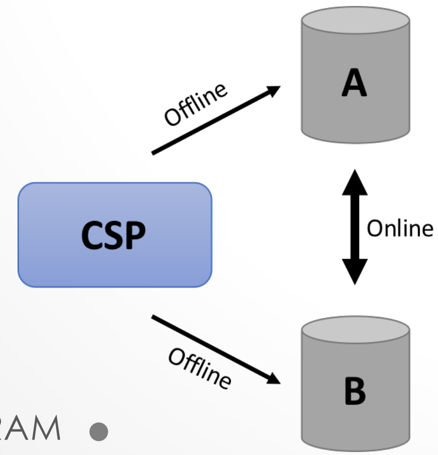


- Each entry of A is a dot product of the vectors held by two different parties
  - In the multi-party case too
- Two party computation of dot product

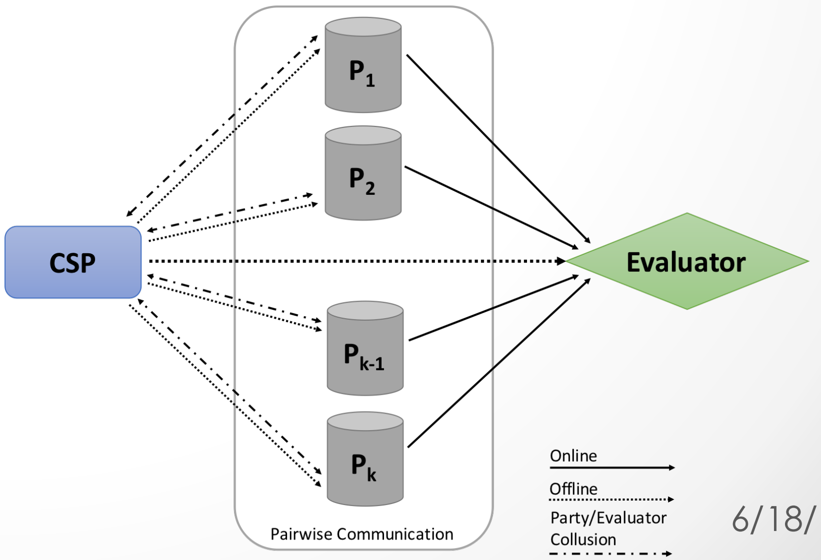
Phase 1

- Architecture – inspired by [NWIJBT13]
  - Two additional **semi-honest, non-colluding** parties:
    - *Crypto Service Provider (CSP)* – generates parameters
    - *Evaluator* – helps for the evaluation of the protocols, has no inputs
- Our setting

Two Parties

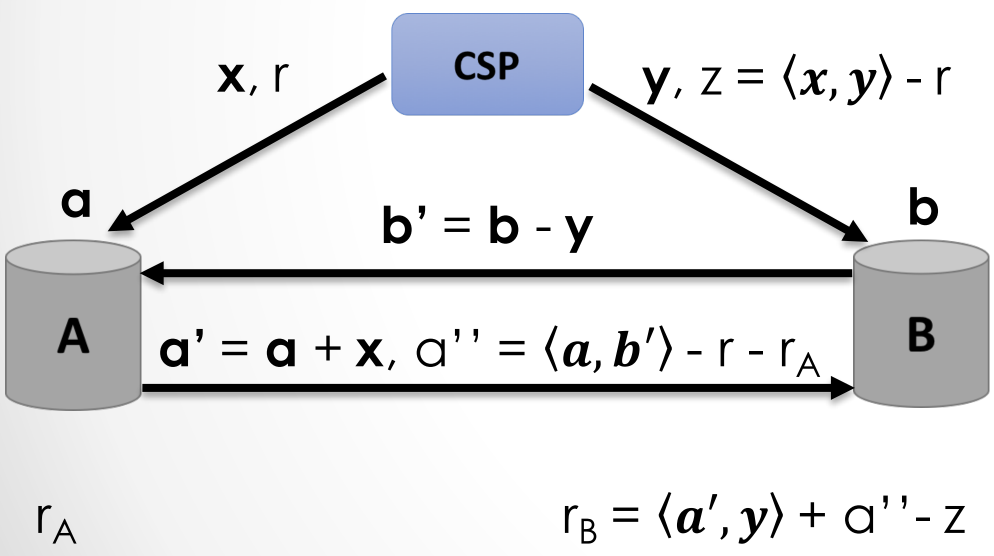


Many Parties

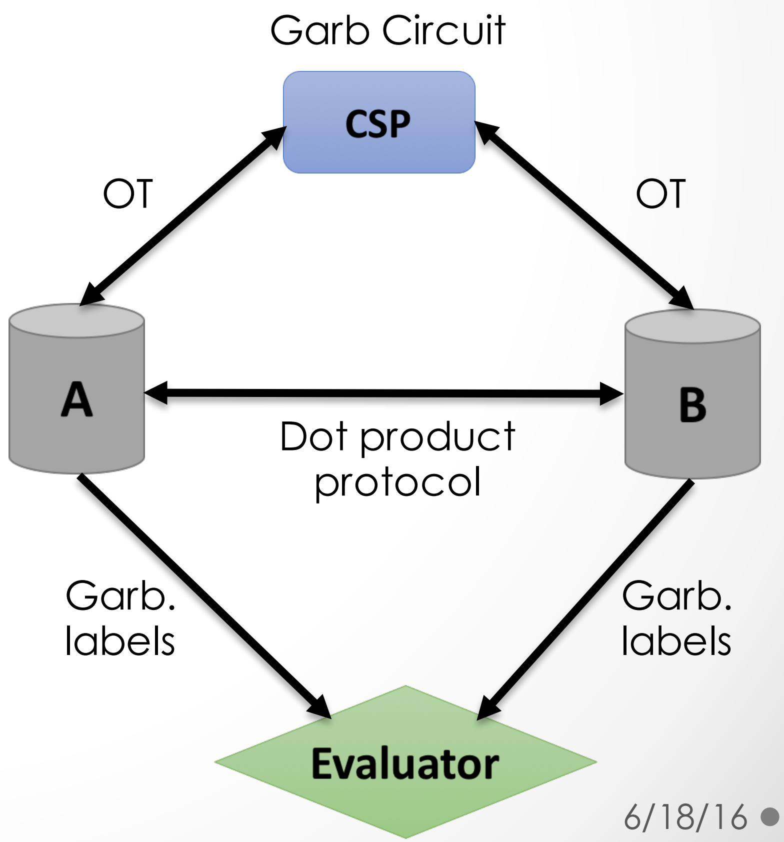


Phase 1

Two Parties



Many Parties



Phase 2

- Two party protocol
  - **Inputs:** additive shares of matrix **A** and vector **b**
  - **Outputs:** additive shares of $\boldsymbol{\theta}$ such that

$$\boldsymbol{A\theta} = \boldsymbol{b}$$

- Gabled circuits computation
- Solutions algorithms
  - Two **exact** algorithms: Cholesky, LDLT
  - One **approximation** algorithm: Conjugate Gradient Descent (CGD)
- [NWIJBT13] implements Cholesky

Cholesky

- *Cholesky decomposition* for positive definite matrices
  - A = LL^T
  - L: d×d lower triangular matrix
- Idea: solve $\mathbf{LL^T}\boldsymbol{\theta} = \boldsymbol{b}$
  - $\mathbf{L}\boldsymbol{\theta}' = \boldsymbol{b}$
  - $\mathbf{L^T}\boldsymbol{\theta} = \boldsymbol{\theta}'$
- Complexity: O(d^3) floating point operations
- Two properties:
  - **Data-agnostic** – no pivoting
  - **Numerically robust** – suitable for finite precision implementations

Algorithm 1: Cholesky

Input : A, b
Output: Solution θ to $A\theta = b$

for $j = 1 \ldots d$ **do**
  $L_{jj} = \boxed{\sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}}$
  for $i = j+1 \ldots d$ **do**
    $L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk})/L_{jj}$
  end
end

$\theta'_1 = b_1/L_{11}$ forward substitution
for $i = 2 \ldots d$ **do**
  $\theta'_i = (b_i - \sum_{j=1}^{i-1} L_{ij}\theta'_j)/L_{ii}$
end

$\theta_d = \theta'_d/L_{dd}$ backward substitution
for $i = d-1 \ldots 1$ **do**
  $\theta_i = (\theta'_i - \sum_{j=i+1}^{d} L_{ji}\theta_j)/L_{ii}$
end

LDLT

- *Variant* of Cholesky decomposition
  - A = LDL^T
  - L – lower triangular
  - D – diagonal, non-negative entries
- Idea: solve $\mathbf{LDL^T}\boldsymbol{\theta} = \boldsymbol{b}$
  - $\mathbf{L}\boldsymbol{\theta}'' = \boldsymbol{b}$
  - $\mathbf{D}\boldsymbol{\theta}' = \boldsymbol{\theta}''$
  - $\mathbf{L^T}\boldsymbol{\theta} = \boldsymbol{\theta}'$
- Complexity: $O(d^3)$
  - No square root
  - Additional substitution phase
- Same properties

Algorithm 2: LDLT

Input : A, b
Output: Solution θ to $A\theta = b$

for $j = 1 \ldots d$ **do**
  $D_j = A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 D_k$
  for $i = j+1 \ldots d$ **do**
    $L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} D_k)/D_j$
  end
end
$\theta'_1 = b_1$
for $i = 2 \ldots d$ **do**
  $\theta'_i = b_i - \sum_{j=1}^{i-1} L_{ij}\theta'_j$
end
$\theta_d = \theta'_d / D_d$
for $i = d-1 \ldots 1$ **do**
  $\theta_i = \theta'_i / D_i - \sum_{j=i+1}^{d} L_{ji}\theta_j$
end

CGD

- Approximate solution
- Solving $A\theta = b$ by solving the optimization

  $$\mathbf{argmin}_{\boldsymbol{\theta}} \, ||\boldsymbol{A\theta} - \boldsymbol{b}||^2$$

- Iterative solutions approach based on conjugate gradients
- Complexity
  - o Until convergence $O(d^3)$
  - o Early termination $O(d^2)$ per iteration
- Error: **ε after $\boldsymbol{O}(\sqrt{\boldsymbol{\kappa}}\,\mathbf{log}\,1/\boldsymbol{\varepsilon})$ iterations**
  - o $\boldsymbol{\kappa}$ - condition number

Algorithm 3: Conjugate Gradient Descent

Input : A, b, number of iterations k
Output: Approximate solution θ_k to $A\theta = b$

Let $\theta_0 = 0$ and $p_0 = r_0 = A\theta_0 - b$
for $t = 0 \ldots k$ **do**
  $\theta_{t+1} = \theta_t - \frac{\langle r_t, r_t \rangle}{p_t^\top A p_t} p_t$
  $r_{t+1} = A\theta_{t+1} - b$
  $p_{t+1} = r_{t+1} + \frac{\langle r_{t+1}, r_{t+1} \rangle}{\langle r_t, r_t \rangle} p_t$
end

Fixed-Point Arithmetic

$$\mathbb{R} \underset{\tilde{\phi}_\delta}{\overset{\phi_\delta}{\rightleftarrows}} \mathbb{Z} \underset{\tilde{\varphi}_q}{\overset{\varphi_q}{\rightleftarrows}} \mathbb{Z}_q$$

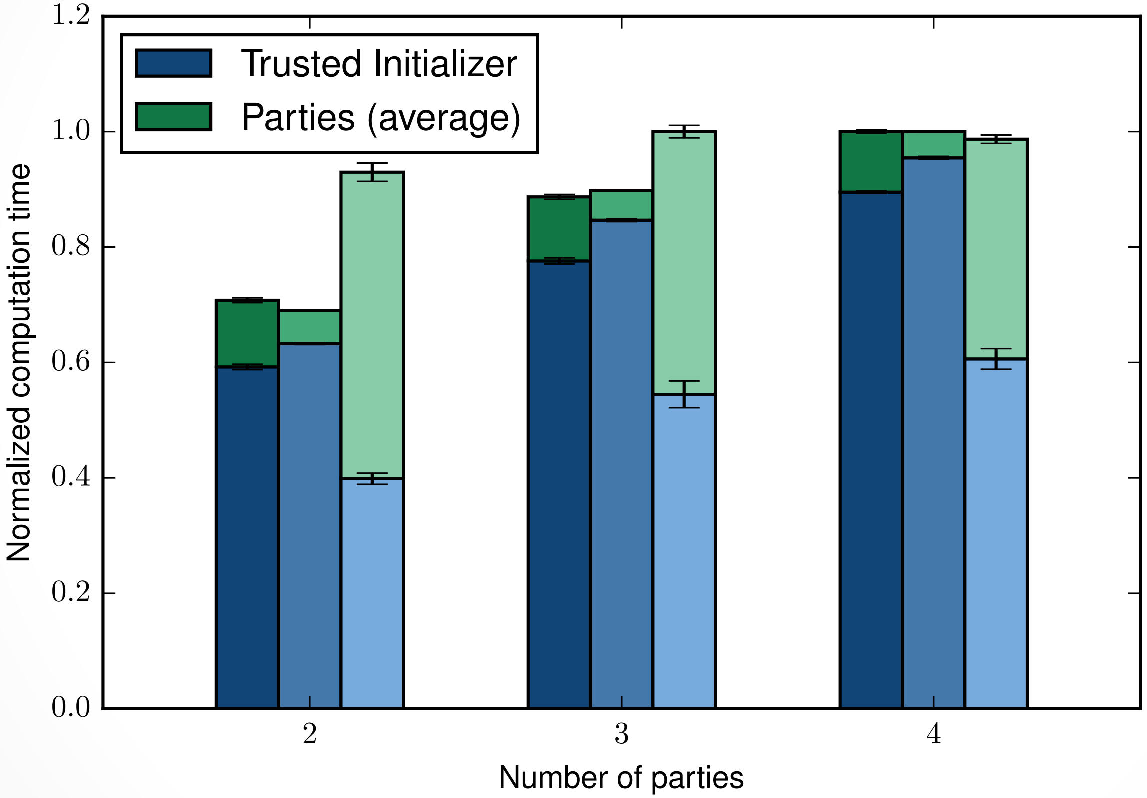
- $\phi_\delta(r) = [r/\delta]$; $\tilde{\phi}_\delta(z) = z\delta$, $|r - \tilde{\phi}_\delta(\phi_\delta(r))| \le \delta$
- $\varphi(z) = z$ if $z \ge 0$; $\varphi(z) = z + q$ if $z < 0$
- $\tilde{\varphi}(u) = u$ if $0 \le u \le q/2$; $\tilde{\varphi}(u) = u - q$ if $q/2 < u \le q - 1$

- Phase1: n-dim vectors with entries of size R
  - Error: $n(2R\delta + \delta^2)$
  - **Normalize** $R \le 1/\sqrt{n} \Rightarrow$ error ε with $\delta = \varepsilon / 2\sqrt{n}$ and $q = 8n/\varepsilon^2$
    - $O(\log(n/\varepsilon))$ bit representation
- Phase2 – experiments
  - $q = 2^{32}$ (4 bits integer part, 1 bit sign) $\Rightarrow \delta = 2^{-27}$
  - $q = 2^{64}$ (4 bits integer part, 1 bit sign) $\Rightarrow \delta = 2^{-59}$

Implementation and Evaluation

- Obliv-C
  - Most recent optimizations: Free XOR, Garbled Row Reduction, Fixed Key Block Ciphers, Half Gates
- Fixed point arithmetic on top of Obliv-C
  - Algorithms: multiplication (Karatsuba-Comba), division (Knuth's algorithm D), square root(Newton's method)
  - 32 bits: 4 bits (integral part) + 28 bit (fractional part)
- Synthetic datasets (vs real datasets)
  - Generated with correct **λ** parameter – sample from d-dimensional Gaussian distribution
  - Tuning **λ** privately is hard question – incorrect **λ** makes the optimization too easy or too difficult
- Amazon EC2 C4 (15GB RAM, 8 CPU cores)

Phase 1

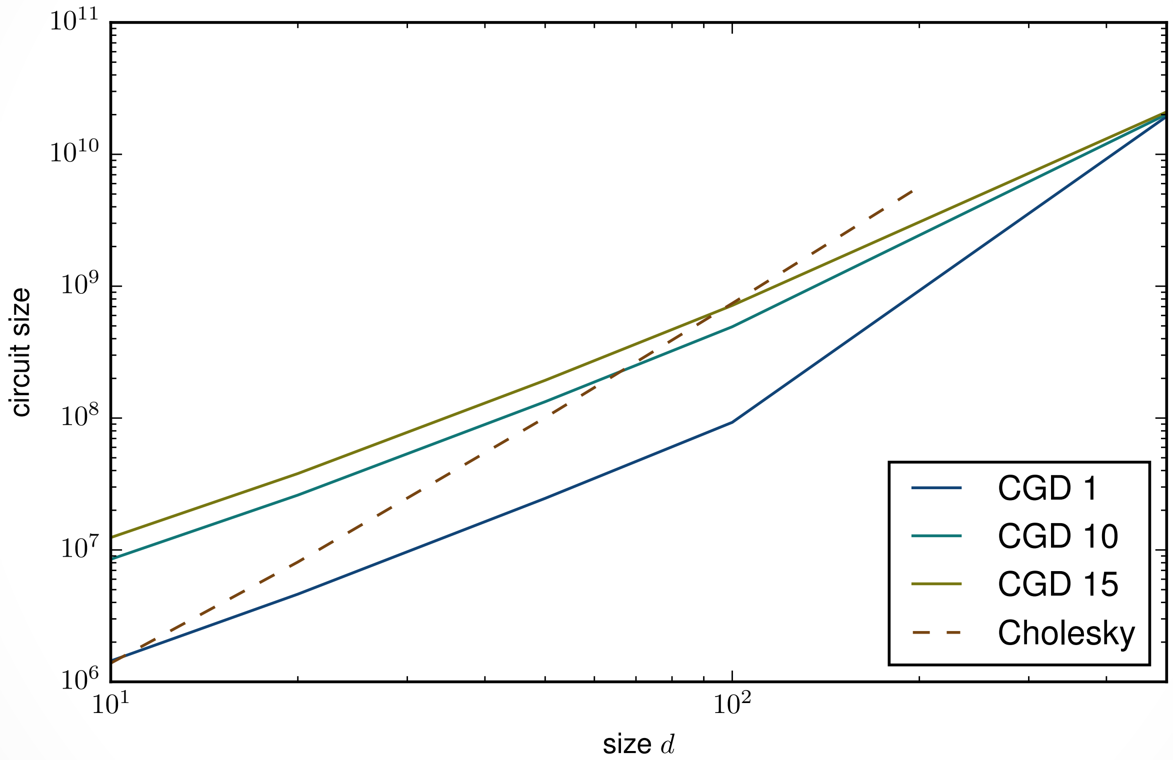


Database partitioned equally among parties

	(n, d)
column1	(2000, 20)
column2	(10000,100)
column3	(50000,500)

	Number of parties					
d	2		3		4	
20	0.17	0.033	0.22	0.032	0.26	0.030
100	19	1.7	26	1.6	29	1.4
500	109	146	149	125	166	104

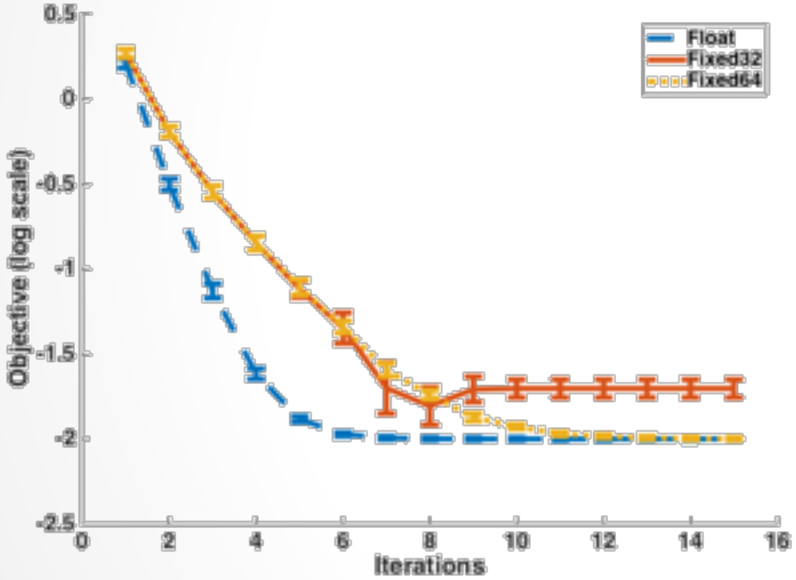
Phase 2



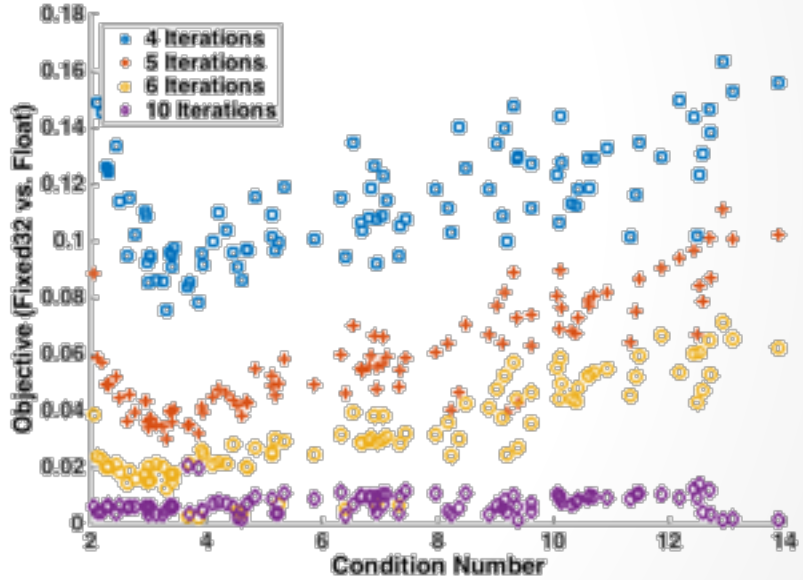
CGD-15	10	20	50	100	500
	2.47668	7.40373	37.8064	140.652	4876.59

Cholesky	10	20	50	100	200
	0.403127	1.69357	20.4527	162.049	1157.95

Phase 2



Convergence of CGD



Fixed vs Floating Point

Conclusions

- Machine learning algorithms – target for MPC
- Ridge regression
  - Vertically partitioned datasets
- Tailored protocol for Phase1
- Two party computation for solving systems of linear equations for Phase2
  - Exact (Cholesky, LDLT) and approximation (CGD) algorithms
  - Approximation: **more efficient with sufficient precision**
- Next steps – classification (logistic regression)

Thank You!