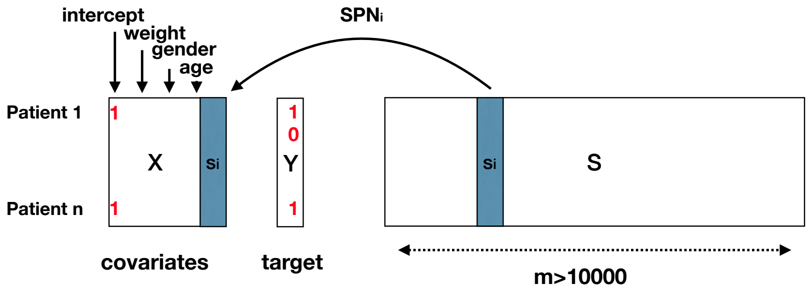
Privacy-preserving Efficient Subset of Features Selection for Regression Models

N. Gama, M. Georgieva

inpher.io

December 10, 2018

GWAS (find the best additional feature)



Question: Is the new future important?

Naive method: compute $stat_i$ for each i...

... that means compute more than 10000 logreg

Description of Idash 2018 Task 2

Goal:

Develop a secure parallel outsourcing solution to compute Genome Wide Association Studies (GWAS) based on linear/logistic regression using **homomorphically encrypted** data.

Challenge (informally):

- Currently: a logreg model, 250 patients, with 3 or 4 physical features.
- Which new feature, among 10000 possible genes (SNP), would improve the model?

Semi-parallel approach

- Don't do 10000 logregs...

Description of Idash 2018 Task 2

inpher.io

Goal:

Develop a secure parallel outsourcing solution to compute Genome Wide Association Studies (GWAS) based on linear/logistic regression using **homomorphically encrypted** data.

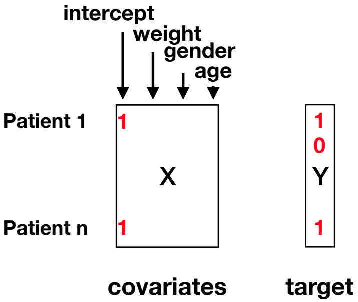
Challenge (informally):

- Currently: a logreg model, 250 patients, with 3 or 4 physical features.
- Which new feature, among 10000 possible genes (SNP), would improve the model?

Semi-parallel approach

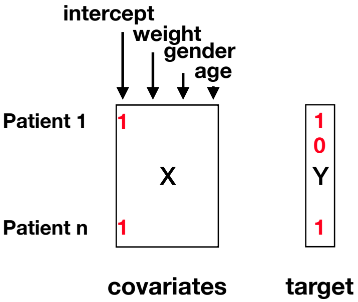
- Don't do 10000 logregs...

Logreg, IRLS, relevance of a feature



- Single Logistic regression:
  - Find θ s.t $Y = sign(X\theta)$
- IRLS:
  - Compute $grad = X^t(Y - p)$, with $p = \sigma(X\theta)$
  - Compute $Hessian = X^t diag(p(1-p))X$

Logreg, IRLS, relevance of a feature



Importance of the i^{th} feature:

- the i^{th} coeff is big: θ_i (numerator)
- the i^{th} error term is small: $(Hess^{-1})_{i,i}$ (denominator)

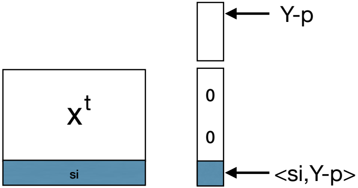
stat= ratio

- Single Logistic regression:
  - Find θ s.t $Y = sign(X\theta)$
- IRLS:
  - Compute $grad = X^t(Y - p)$, with $p = \sigma(X\theta)$
  - Compute $Hessian = X^t diag(p(1-p))X$

Semi-parallel GWAS (optimized)

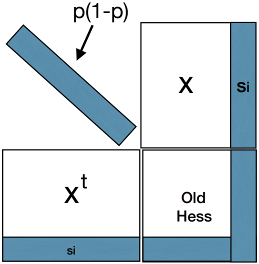
1. Do $logreg(X, y)$ without S
2. Once model is converged, add s_i

- Gradient:



They can be batch-computed: $(Y\text{-}p)^t S$

- Hessian:



MPC versus FHE

inpher.io

FHE

- Long term storage
- Unique Cloud
- Slower and consumes more memory

MPC

- Faster than FHE
- More accuracy
- All data owner must participate

Fixed points versus Floating point

Floating point:

- $x = m.2^{\tau}$, with $m \in 2^{-\rho}.\mathbb{Z}$ and $\frac{1}{2} \leq |m| < 1$
- $\tau = \lceil log_2(x) \rceil$ data dependent and not public (not FHE-friendly)
- The exponent is always in sync with the data
  ex: $(1.23 \cdot 10^{-4}) * (7.24 \cdot 10^{-4}) = (8.90 \cdot 10^{-8})$

Fixed point:

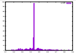
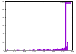
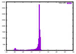
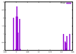
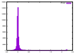
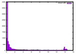
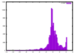
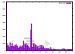
- $x = m.2^{\tau}$, with $m \in 2^{-\rho}.\mathbb{Z}$ and $0 \leq |m| < 1$,
- τ is public, thus FHE-friendly
- Risk of overflow (τ too small)
- Risk of underflow (τ too large)
  ex: $(0.000123 \cdot 10^{0}) * (0.000724 \cdot 10^{0}) = (0.000000 \cdot 10^{0})$

Plaintext parameters:

- $\rho \in \mathbb{N}$: bits of precision of the plaintext (≈ 15 bits)
- $\tau \in \mathbb{Z}$: slot exponent (order of magnitude of the complex values in each slot)

Choice of slot exponent

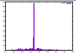
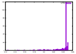
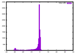
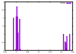
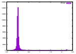
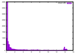
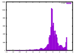
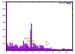
The slot exponent τ that defines the plaintext interval must be carefully estimated.

variable	avg	stdev	min	max	dist
p	0.440816	0.0975715	0.176397	0.853487	
w	0.236977	0.0201871	0.125047	0.25	
z_i^*	-3.33092	7.36068	-30.9426	31.2008	
G	0.0577846	0.0953495	-0.011997	0.236977	
A	0.0621965	0.301255	-0.317312	2.236	
$(s_i^*)^2$	2.44243	4.11085	0.111961	14.5044	
$\log(stat_i)$	0.200039	1.84459	-13.7207	4.36158	
$p-value$	0.310218	0.24083	0	0.999163	

Choice of slot exponent

inpher.io

The slot exponent τ that defines the plaintext interval must be carefully estimated.

variable	avg	stdev	min	max	dist
p	0.440816	0.0975715	0.176397	0.853487	
w	0.236977	0.0201871	0.125047	0.25	
z_i^*	-3.33092	7.36068	-30.9426	31.2008	
G	0.0577846	0.0953495	-0.011997	0.236977	
A	0.0621965	0.301255	-0.317312	2.236	
$(s_i^*)^2$	2.44243	4.11085	0.111961	14.5044	
$\log(stat_i)$	0.200039	1.84459	-13.7207	4.36158	
$p-value$	0.310218	0.24083	0	0.999163	

Numerical stability

inpher.io

Not stable

Increase the precision of the algorithm, but that implies bigger parameters.

Stable

Use stable computation with negative feedback
(e.g. gradient descent)
Smaller parameters

FHE Solution

inpher.io

FHE parameters:

- $L \in \mathbb{N}$: level exponent of the ciphertext ($\alpha = 2^{-(L+\rho)}$: noise rate)
- $N = f(\lambda, \alpha)$: key size, with λ the security parameter

The `lwe-estimator` script was used to assert the security.
(conform to HE security standardization white paper)

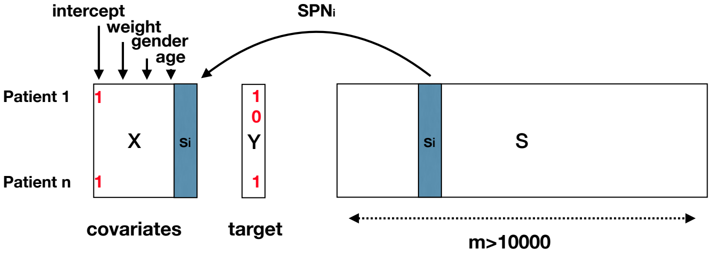
FHE Solution

FHE parameters:

- $L \in \mathbb{N}$: level exponent of the ciphertext ($\alpha = 2^{-(L+\rho)}$: noise rate)
- $N = f(\lambda, \alpha)$: key size, with λ the security parameter

The `lwe-estimator` script was used to assert the security.
(conform to HE security standardization white paper)

Plaintext algorithm in FHE solution



Input:

- $X \in \mathcal{M}_{n,k+1}(\mathbb{R})$ input matrix
- $y \in \mathbb{B}^n$ binary vector
- $S \in \mathcal{M}_{n,m}(\mathbb{R})$ assumed binary

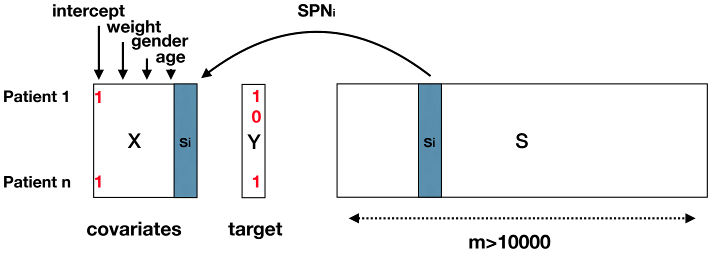
Output:

- $stat \in \mathbb{R}^m$
  with $stat_i = \frac{\mathbf{z}^*{}_i}{\sqrt{\mathbf{s}^*{}_i^2}}$

Key points of our solution:

- Make plaintext algorithm FHE friendly
- Use hybrid homomorphic encryption

Plaintext algorithm in FHE solution



Input:

- $X \in \mathcal{M}_{n,k+1}(\mathbb{R})$ input matrix
- $y \in \mathbb{B}^n$ binary vector
- $S \in \mathcal{M}_{n,m}(\mathbb{R})$ assumed binary

Output:

- $stat \in \mathbb{R}^m$
  with $stat_i = \frac{\mathbf{z^*}_i}{\sqrt{\mathbf{s^*}_i^2}}$

Key points of our solution:

- Make plaintext algorithm FHE friendly
- Use hybrid homomorphic encryption

Optimization of plaintext algorithm

Make the plaintext algorithm FHE friendly

- Find simple geometric equivalents of the formula
- Find approximation with lower multiplicative depth
- Replace feature scaling of X with orthogonalization

Algorithm in plaintext

Algorithm 2 Plaintext algorithm

$\beta^{(0)} = (0, \dots, 0)$
for $t = 1$ to iters **do**
  $\beta^{(t)} \leftarrow \beta^{(t-1)} + \text{step} \cdot X^T. \left(\mathbf{y} - \sigma(X\beta^{(t-1)})\right)$
end for ▷ logreg
$\mathbf{p} \leftarrow \sigma(X\beta^{(\text{iters})})$
$\mathbf{z}^* \leftarrow (\mathbf{y} - \mathbf{p})^T.S$ ▷ numerator
$W \leftarrow \text{diag}(p * (1-p))$
$G \leftarrow X^T.W.X \approx \frac{1}{4} * Id$ (assumed that X orthogonal)
$A \leftarrow X^T.W.S$
$\mathbf{s}^{*\mathbf{2}} = \text{colsums}(W \cdot (S \odot S)) - \text{colsums}(A \odot G^{-1}A)$
$(\approx A[0] * \sqrt{n} - 4 * \text{colsums}(A \odot A))$ ▷ denominator
$\mathbf{r}_i = [2 \cdot \log(|\mathbf{z}^*{}_i|) - \log(|\mathbf{s}^{*\mathbf{2}}{}_i|)]$ for each $i \in [1, m]$ ▷ log of stat

$$stat_i = \frac{\mathbf{z}^*{}_i}{\sqrt{\mathbf{s}^*{}_i^2}} = \frac{1}{2}\exp(\mathbf{r}_i)$$

$$\text{p-value}_i = \text{p-Norm}(stat_i)$$

Algorithm in plaintext

inpher.io

Algorith

for loops
(better with fast bootstrapping)

$\beta^{(0}$

for $t = 1$ to iters **do**

$\quad \beta^{(t)} \leftarrow \beta^{(t-1)} + \text{step} \cdot X^T . \left(\mathbf{y} - \sigma(X\beta^{(t-1)}) \right)$

enc ▷ logreg

continuous non-polynomial functions
(Approx numbers, or Lookup tables)

$\mathbf{p}$

$\mathbf{z}^*$ ▷ numerator

$W \leftarrow \text{diag}(p * (1 - p))$

$G \leftarrow X^T.W.X \approx \frac{1}{4} * Id$ (assumed that X orthogonal)

$A \leftarrow X^T.W.S$

$\mathbf{s}^{*\mathbf{2}} = \text{colsums}(W \cdot (S \odot S)) - \text{colsums}(A \odot G^{-1} A)$

$(\approx A[0] * \sqrt{n} - 4 * \text{colsums}(A \odot A))$ ▷ denominator

$\mathbf{r}_i = [2 \cdot \log(|\mathbf{z}^*{}_i|) - \log(|\mathbf{s}^{*\mathbf{2}}{}_i|)]$ for each $i \in [1, m]$ ▷ log of stat

$$stat_i = \frac{\mathbf{z}^*{}_i}{\sqrt{\mathbf{s}^{*}{}_i^{2}}} = \frac{1}{2}\exp(\mathbf{r}_i)$$

$$\text{p-value}_i = \text{p-Norm}(stat_i)$$

Algorithm in plaintext

Algorith

for loops
(better with fast bootstrapping)

$\beta^{(0}$

for $t = 1$ to iters **do**

$\quad \beta^{(t)} \leftarrow \beta^{(t-1)} + \text{step} \cdot X^T . \left(\mathbf{y} - \sigma(X\beta^{(t-1)})\right)$

enc

individual non-linear operations in small dimension
(lookup tables)

$\mathbf{p}$

continuous non-polynomial functions
(Approx numbers, or Lookup tables)

$\mathbf{z}^*$ ▷ numerator

$W \leftarrow$ diag(

$G \leftarrow X^T.$ … nal)

multiplication with fresh ciphertexts
(better with TFHE's external product)

$A \leftarrow X^T.W.S$

$\mathbf{s^{*2}} = \text{colsums}(W \cdot (S \odot S)) - \text{colsums}(A \odot G^{-1}A)$

$(\approx A[0] * \sqrt{n} - 4 * \text{colsums}(A \odot A))$ ▷ denominator

$\mathbf{r}_i = [2 \cdot \log(|\mathbf{z^*}_i|) - \log(|\mathbf{s^{*2}}_i|)]$ for each $i \in [1, m]$ ▷ log of stat

$$stat_i = \frac{\mathbf{z^*}_i}{\sqrt{\mathbf{s^*}_i^2}} = \frac{1}{2}\exp(\mathbf{r}_i)$$

$$\text{p-value}_i = \text{p-Norm}(stat_i)$$

Algorithm in plaintext

Algorith

for loops
(better with fast bootstrapping)

1: $\beta^{(0)}$
2: **for** $t = 1$ to iters **do**
3: $\beta^{(t)} \leftarrow \beta^{(t-1)} + \text{step} \cdot X^T . \left(\mathbf{y} - \sigma(X\beta^{(t-1)})\right)$
4: **en**

individual non-linear operations in small dimension
(lookup tables)

continuous non-polynomial functions
(Approx numbers, or Lookup tables)

5: $\mathbf{p}$
6: $\mathbf{z}^*$ ▷ numerato
7: $W \leftarrow$ diag(
8: $G \leftarrow X^T.$ nal)

multiplication with fresh ciphertexts
(better with TFHE's external product)

9: $A \leftarrow X^T.W.S$
10: $\mathbf{s^{*2}} = \text{colsums}(W \cdot (S \odot S)) - \text{colsums}(A \odot G^{-1}A)$
11: $(\approx A[0] * \sqrt{n} - 4 * \text{colsums}(A \odot A))$
12: $\mathbf{r}_i = [2 \cdot \log(|\mathbf{z^*}_i|) - \log(|\mathbf{s^{*2}}_i|)]$ for each $i \in [1, m]$ ▷ log of stat

very large dimension
(fully packed SIMD)

$$stat_i = \frac{\mathbf{z^*}_i}{\sqrt{\mathbf{s^*}_i^2}} = \frac{1}{2}\exp(\mathbf{r}_i)$$

$$\text{p-value}_i = \text{p-Norm}(stat_i)$$

continuous function batched on a large vector

Algorithm in plaintext

inpher.io

Algorith

for loops
(better with fast bootstrapping)

1: $\beta^{(0)}$
2: **for** $t = 1$ to iters **do**
3: $\beta^{(t)} \leftarrow \beta^{(t-1)} + \text{step} \cdot X^T . \left(\mathbf{y} - \sigma(X\beta^{(t-1)})\right)$
4: **end**

continuous non-polynomial functions
(Approx numbers, or Lookup tables)

individual non-linear operations in small dimension
(lookup tables)

5: $\mathbf{p}$
6: $\mathbf{z}^*$ ▷ numerator
7: $W \leftarrow$ diag
8: $G \leftarrow X^T.$ nal)

multiplication with fresh ciphertexts
(better with TFHE's external product)

9: $A \leftarrow X^T.W.S$
10: $\mathbf{s}^{*\mathbf{2}} = \text{colsums}(W \cdot (S \odot S)) - \text{colsums}(A \odot G^{-1}A)$
11: $(\approx A[0] * \sqrt{n} - 4 * \text{colsums}(A \odot A))$
12: $\mathbf{r}_i = [2 \cdot \log(|\mathbf{z}^*{}_i|) - \log(|\mathbf{s}^{*\mathbf{2}}{}_i|)]$ for each $i \in [1, m]$ ▷ log of stat

very large dimension
(fully packed SIMD)

$$stat_i = \frac{\mathbf{z}^*{}_i}{\sqrt{\mathbf{s}^{*2}_i}} = \frac{1}{2}\exp(\mathbf{r}_i)$$

$$\text{p-value}_i = \text{p-Norm}(stat_i)$$

continuous function batched on a large vector

Which fully homomoprhic scheme should we choose?

Each library has its own strengths

inpher.io

Strengths of HE libraries

- BGV/Helib: SIMD finite field arithmetic
- B/FV, Seal: SIMD vector $\mod t$
- HEAAN: SIMD fixed point arithmetic
- TFHE: single evaluation, boolean logic, comparison, threshold, complex circuits
- etc...

How to get all the benefits without the limitations?

Solution: Chimera

inpher.io

Idea:

- Unified plaintext space over the Torus
- Switch between ciphertext representations
- Implement bridges between TFHE, B/FV and HEAAN

For this use-case

We use the switch between TFHE and HEAAN!

Solution: Chimera

inpher.io

Idea:

- Unified plaintext space over the Torus
- Switch between ciphertext representations
- Implement bridges between TFHE, B/FV and HEAAN

For this use-case

We use the switch between TFHE and HEAAN!

Chimera solution

1. Initial Logreg on matrix X and vector y
    - adapt lib TFHE + logreg
2. Mass Linear algebra computations
    - implement Chimera (version 2 of TFHE)
3. Batch Logarithm computation
    - adapt lib HEAAN

Benchmarks (Idash Bootstrapped)

inpher.io

Steps	Timing (4 cores)	Timing (96 cores)	RAM
KeyGen	5.5 mins	2.0 mins	4.4 GB
Encryption	7.2 mins	1.3 mins	8.6 GB
Cloud Computation	3h06	10.2 mins	7.8 GB

- Input ciphertext: 5GB (enc X, y, S)
- Final ciphertext: 640KB (enc numerator + denominator)

Benchmarks (with new optimizations)

inpher.io

$k = 3,\ n = 250,\ m = 10000$

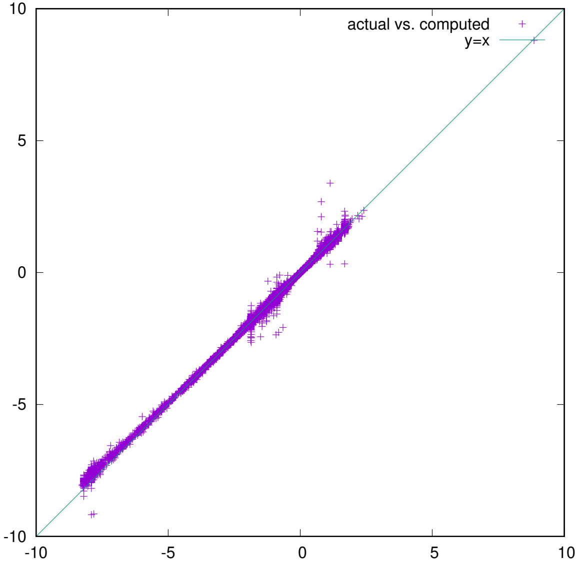
Steps	**Timing (4 cores)**	**Timing (96 cores)**	**RAM**
KeyGen	5.5 mins	2.0 mins	4.4 GB
Encryption	7.2 mins	1.3 mins	8.6 GB
Cloud Computation	35 mins	3 mins	7.8 GB

$k = 7,\ n = 250,\ m = 10000$

Steps	**Timing (4 cores)**	**Timing (96 cores)**	**RAM**
KeyGen	5.5 mins	2.0 mins	4.4 GB
Encryption	7.2 mins	1.3 mins	8.6 GB
Cloud Computation	41 mins	3.1 mins	7.8 GB

- initial ciphertext: 5GB (enc X, y, S)
- final ciphertext: 640KB (enc numerator + denominator)

Numerical Accuracy (FHE has noise)



Questions?