Improved Privacy-Preserving Training using fixed-Hessian Minimisation

Tabitha Ogilvie, Rachel Player, Joe Rowell

Information Security Group

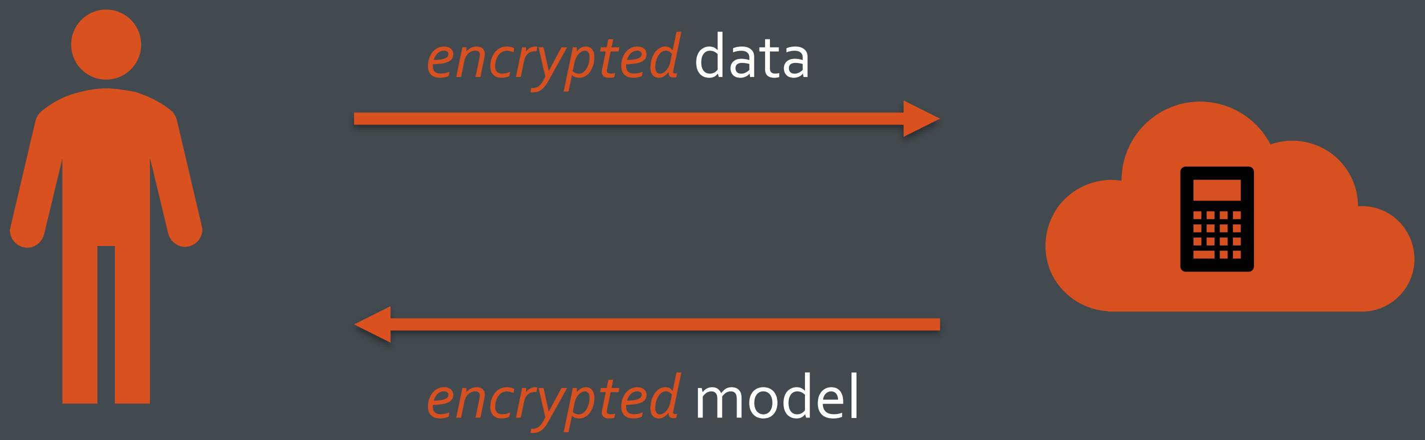
WAHC 2020

Contributions

- A new method for homomorphic Logistic Regression training using CKKS
- A comparison of 3 different Logistic Regression training methods at 128-bit security
- An improved method for homomorphically calculating $\frac{1}{x}$
- Three methods for homomorphic Ridge Regression training

Background


data
model



encrypted data
encrypted model

Homomorphic Encryption

A scheme E is fully homomorphic if, given a function f and an encryption of some data $E(x)$, we can generate $E(f(x))$ *without decrypting*

Models

Logistic Regression

- A binary classifier based on the sigmoid function
- No closed form solution
- Cost function given by

$$J(\beta) = -\sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^T \mathbf{x}_i))$$

Ridge Regression

- Linear regression with L2 regularisation
- Closed form solution
- Cost function given by

$$J(\beta) = \frac{1}{2}(\lambda \sum_{i=1}^{d} \beta_i^2 + \sum_{i=1}^{n} (y_i - \beta^T x_i)^2)$$

Prior Work

Main Comparisons

Kim18a achieves Logistic Regression training using:

- "Feature wise" encoding
- 80-bit security
- Gradient Descent

$$\beta^{(k+1)} = \beta^{(k)} - \alpha \nabla J\left(\beta^{(k)}\right)$$

Kim18b achieves Logistic Regression training using:

- "database" encoding
- 80-bit security
- Nesterov's Accelerated Gradient Descent

$$\begin{cases} \beta^{(k+1)} & = v^{(k)} - \alpha_k \cdot \nabla J(v^{(k)}) \\ v^{(k+1)} & = (1-\gamma_k) \cdot \beta^{(k+1)} + \gamma_k \cdot \beta^{(k)} \end{cases}$$

Kim18a: Kim, M., *et al.* (2018). Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics, 6(2), e19.

Kim18b: Kim, A., et al. (2018). Logistic regression model training based on the approximate homomorphic encryption. BMC medical genomics, 11(4), 83.

Hessian-Based Minimisation

ROYAL HOLLOWAY UNIVERSITY OF LONDON

$$\beta^{(k+1)} = \beta^{(k)} - H(\beta^{(k)})^{-1} \nabla J(\beta^{(k)})$$

- $H(\beta^{(k)})$ depends on the current value of the parameters
- We need to invert a matrix

Böhning and Lindsay: replace the full Hessian with a *fixed* symmetric positive definite matrix $\widetilde{H}$ such that $\widetilde{H} \geq H$:

- Monotonic
- Guaranteed convergence to a minimum (if J is bounded below)
- Linear rate of convergence

D. Böhning and B. Lindsay. 1988. Monotonicity of quadratic-approximation algorithms. Annals of the Institute of Statistical Mathematics40 (02 1988), 641–663.

Fixed Hessian Logistic Regression Training

$$H(\beta^{(k)}) \xrightarrow{\text{BL88}} \frac{1}{4}X^T X \xrightarrow{\text{BV18: row sums}} \overline{H}$$

D. Böhning and B. Lindsay. 1988. Monotonicity of quadratic-approximation algorithms. Annals of the Institute of Statistical Mathematics40 (02 1988), 641–663.
C. Bonte and F. Vercauteren. 2018. Privacy-preserving logistic regression training. BMC medical genomics 11, Suppl 4 (October 2018), 86.

Our Work

Logistic Regression

Update the BV fixed-Hessian approach, adding the following features:

- Use the CKKS encoding to parallelise the calculation of the entries of $\overline{H}$, consuming only 1 level
- Use a data pre-processing step similar to 18a to reduce the number of levels per parameter update to 3
- Replace the linear Taylor approximation with a linear Chebyshev approximation
- To invert the entries $\frac{1}{H_{kk}}$ we first use the linear approximation from Schulte et al., and then iterate Newton Raphson 3 times, reducing the relative error

M. J. Schulte, J. Omar, and E. E. Swartzlander. 1994. Optimal initial approximations for the Newton-Raphson division algorithm.Computing 53, 3-4 (1994), 233–242.

Implementation & Comparison

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Descent	Enc	$\log \Delta$	deg g	μ	Time	AUC	Acc. (%)
GD [Kim18a]	F	27	3	8	102.18s	0.89	82.53
	F	28	3	7	59.12s	0.93	83.65
	F	31	3	7	58.62s	0.95	80.46
NAD [Kim18b]	D	30	3	5	42.51s	0.96	88.90
	D	31	3	5	42.57s	0.94	88.74
FH (Ours)	F	39	1	5	45.70s	0.94	88.50
	F	40	1	4	27.18s	0.92	88.26
	F	45	1	4	30.98s	0.92	88.26

Why Use a Fixed-Hessian Method?

$$\beta^{(k+1)} = \beta^{(k)} - \alpha \nabla J \left(\beta^{(k)} \right)$$

$$\begin{cases} \beta^{(k+1)} & = v^{(k)} - \alpha_k \cdot \nabla J(v^{(k)}) \\ v^{(k+1)} & = (1 - \gamma_k) \cdot \beta^{(k+1)} + \gamma_k \cdot \beta^{(k)} \end{cases}$$

- With the Fixed-Hessian method, no step size needs to be chosen
- Adopting “real world” methods for choosing the step size could compromise security

Ridge Regression

- Use the same three minimisation techniques
- Use the BV diagonalization method to approximate the Hessian
- Use a feature-by-feature encoding to parallelise over the size of the dataset
- Use a server-side pre-processing step to reduce computation per iteration

Implementation & Comparison

Descent	$\log \Delta$	μ	Time	r^2
GD	40	9	207.27s	0.4165
	30	13	450.53s	0.4058
NAD	40	6	127.71s	0.4054
	30	9	415.54s	$\boxed{0.4566}$
FH	40	6	137.68s	0.3206

Further Work

- Ridge Regression closed form solution

- CKKS precision
  - Setting the precision factor
  - Precision loss and iterative descent methods
  - Bootstrapping

Thank You!

ROYAL HOLLOWAY UNIVERSITY OF LONDON