Secure Learning of Logistic Regression Model

Homomorphic Evaluation of Gradient Descent

UC San Diego

Jung Hee Cheon[†] & Andrey Kim[†] & Miran Kim[*] & Keewoo Lee[†] & Yongsoo Song[†]

[†] Seoul National University, [*] University of California, San Diego

{jhcheon, kimandrik, activecondor, lucius05}@snu.ac.kr & mrkim@ucsd.edu

HE for Arithmetic of Approximate Numbers [CKKS17]

- The noise of a (Ring) LWE sample is considered as a part of computational error

$$\mathsf{ct} = \mathsf{Enc}_{\mathsf{sk}}(m) \quad \Longrightarrow \quad \langle \mathsf{ct}, \mathsf{sk} \rangle = m + e \pmod q, \quad \textit{i.e.,} \quad [\langle \mathsf{ct}, \mathsf{sk} \rangle]_q \approx m$$

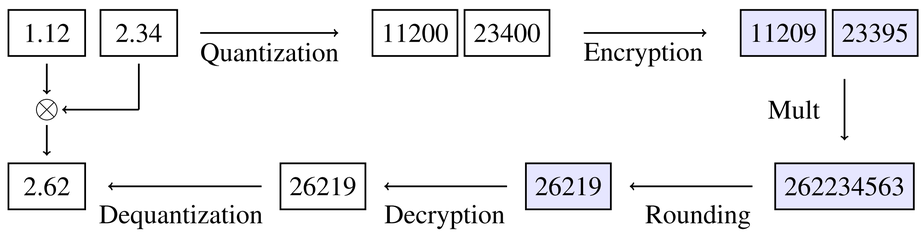
- Perform the approximate addition and multiplication over encrypted data.

$$\mathsf{ct}_1 = \mathsf{Enc}_{\mathsf{sk}}(m_1), \ \ \mathsf{ct}_2 = \mathsf{Enc}_{\mathsf{sk}}(m_2) \quad \Longrightarrow \quad [\langle \mathsf{ct}_{\mathsf{add}}, \mathsf{sk} \rangle]_q \approx m_1 + m_2, \quad [\langle \mathsf{ct}_{\mathsf{mult}}, \mathsf{sk} \rangle]_q \approx m_1 \cdot m_2$$

- Support the homomorphic rounding of plaintexts (*rescaling*).

$$[\langle \mathsf{ct}, \mathsf{sk} \rangle]_q \approx m \quad \Longrightarrow \quad [\langle \mathsf{ct}', \mathsf{sk} \rangle]_{b^{-1} \cdot q} \approx b^{-1} \cdot m$$

- Leveled HE scheme with a linear bitsize $O(L)$ of ciphertext modulus (L: the depth of a circuit)
- Example:



- Provide an open-source implementation (`HEAAN`) in C++ language.
  Available at github (`https://github.com/kimandrik/HEAAN`).

Functionality of `HEAAN`

- Construction over the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ of a power-of-two dimension N.
- Packing multiple numbers (max. $N/2$) in a single ciphertext.
- Slot-wise addition, multiplication, rounding in a SIMD manner.
- Permutation (rotation) on plaintext slots.

Encoding of Database

	Class (± 1)	Feature Vector ($\mathbb{R}^d$)
User 1	$y_1 = 1$	$x_{11} = 2.78 \ x_{12} = 1.12 \ \cdots \ x_{1d} = 3.05$
User 2	$y_2 = -1$	$x_{21} = 0.56 \ x_{22} = 1.58 \ \cdots \ x_{2d} = 2.95$
$\vdots$	$\vdots$	$\vdots \quad \vdots \quad \ddots \quad \vdots$
User n	$y_n = 1$	$x_{n1} = 1.22 \ x_{n2} = 2.01 \ \cdots \ x_{nd} = 4.31$

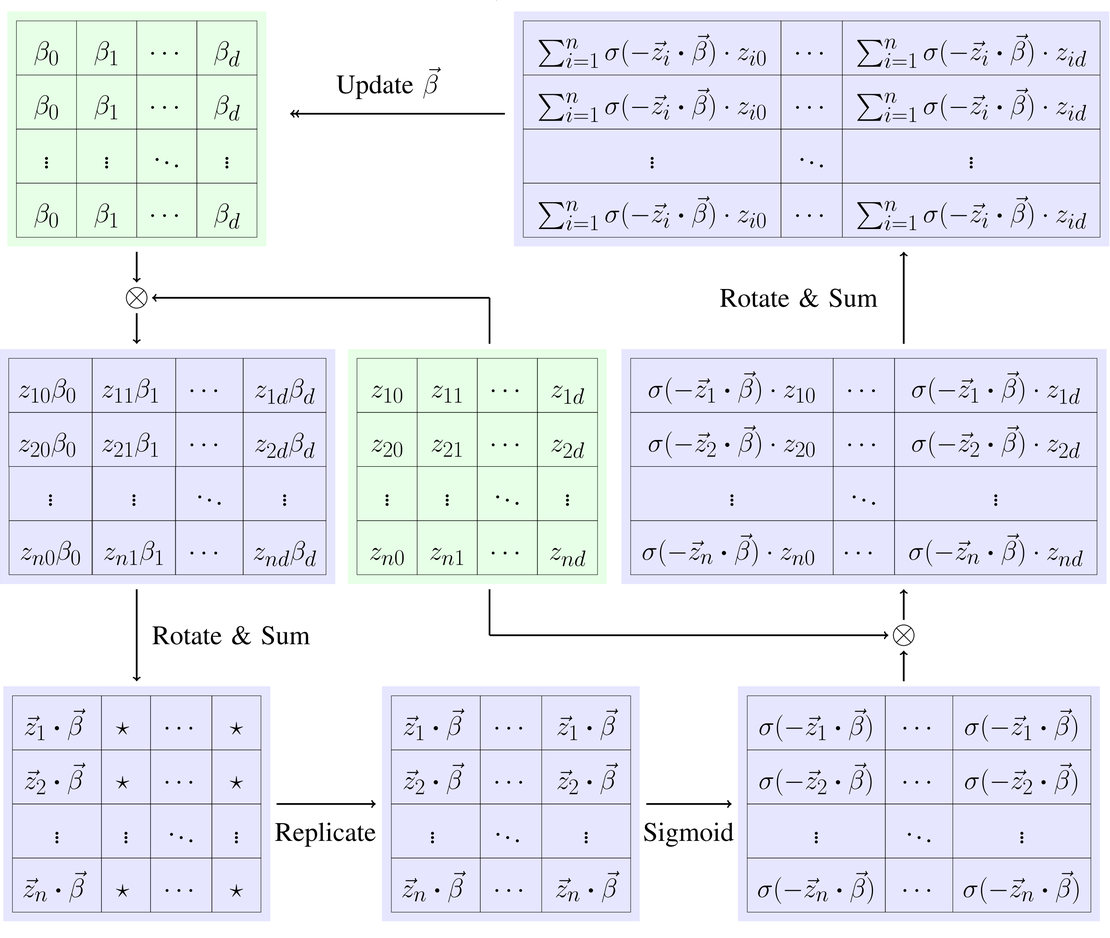
$$\Longrightarrow \ Z = \begin{bmatrix} 1 & 2.78 & 1.12 & \cdots & 3.05 \\ -1 & -0.56 & -1.58 & \cdots & -2.95 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1.22 & 2.01 & \cdots & 4.31 \end{bmatrix}$$

- Write $\vec{z}_i = y_i \cdot (1, \vec{x}_i) \in \mathbb{R}^{d+1}$ for $i = 1, 2, \ldots, n$.
- Generate an encryption of $Z = (\vec{z}_i)_i$.

Homomorphic Evaluation of Gradient Descent

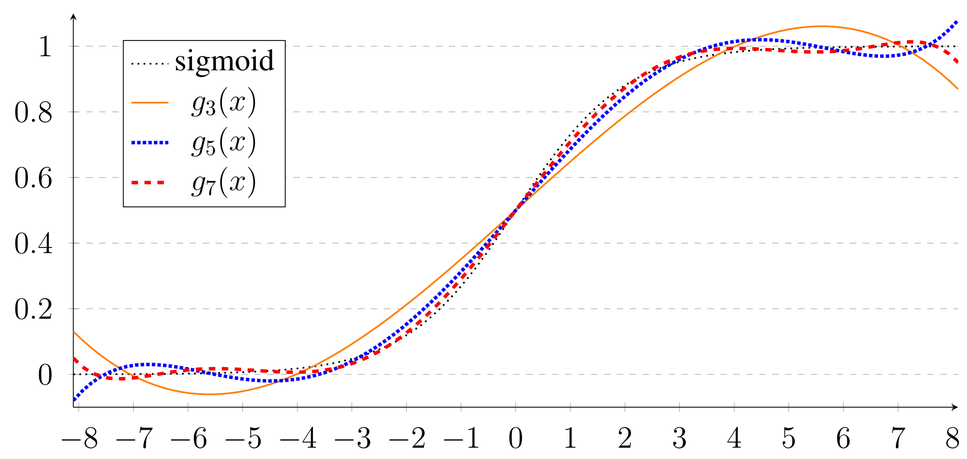
- Our goal is to find a modeling vector $\vec{\beta}$ that minimizes the loss $J(\vec{\beta}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-\vec{z}_i \cdot \vec{\beta}))$.
- The gradient descent algorithm computes the following equation recursively to update a modeling vector and find a better point:

$$\vec{\beta} \leftarrow \vec{\beta} + \frac{\alpha}{n} \sum_{i=1}^{n} \sigma(-\vec{z}_i \cdot \vec{\beta}) \cdot \vec{z}_i$$

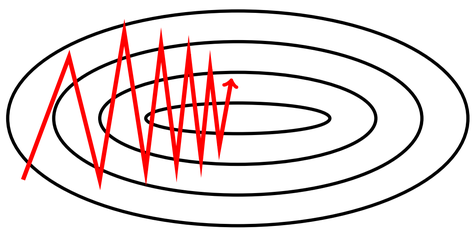


Implementation & Optimization

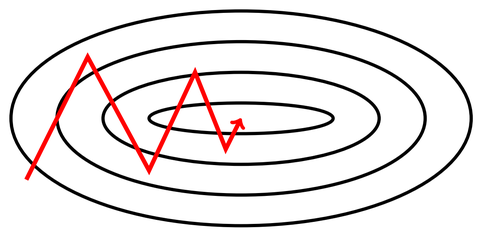
- Choice of polynomials: use the least squares method to find a global approximation of the sigmoid function. The maximum error of the degree 3, 5 and 7 approximations are about 0.114, 0.061 and 0.032, respectively, over the range $[-8, 8]$.



- Standard GD has a navigation problem when the surface curves are much more steep in one dimension than in another, which are common around local optima. Momentum is a method that accelerates GD in the relevant direction and dampens oscillations. It adds a fraction of the update vector of the past time step to the current update vector.



Standard GD



Momentum method

- Nesterov accelerated gradient [Nes83] is a slightly different version of the momentum update. It guarantees a better rate of convergence $O(1/t^2)$ (vs. $O(1/t)$ of standard GD) for convex functions theoretically, and also consistently works slightly better in practice. Instead of evaluating gradient at the current position, the momentum method gives us an approximation of the next position. We therefore evaluate the gradient at this "looked-ahead" position. The updated equations are as follows:

$$\vec{v}_t = \gamma \cdot \vec{v}_{t-1} + \alpha \cdot \bigtriangledown J(\vec{\beta}_{t-1} - \gamma \vec{v}_{t-1})$$
$$\vec{\beta}_t = \vec{\beta}_{t-1} - \vec{v}_t$$

Experimental Results

Performance description in terms of complexity & storage with 80-bit security on a machine with Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz processor.

Parameters				**Results**					
Iter	**Approx Degree**	$\log N$	$\log q$	**Enc Time**	**Learn Time**	**PK Size**	**Encrypted Database**	**Accuracy**	**AUC**
7	5	16	1229	0.95min	10.25min	1.2GB	38 MB	815 / 1421	0.709
14	5	17	2405	2.33min	63.85min	2.4GB	75 MB	888 / 1421	0.715

#**(Features)** = 19, #**(Samples)** = 1421

Conclusion

In this work, we employed the technology of homomorphic encryption to securely evaluate the learning phase of logistic regression method. Our contributions for achieving a better performance can be summarized as follows: (1) we adapt the "`HEAAN` library for an efficient arithmetic of real numbers, (2) use the packing method to reduce the size of encrypted data and the complexity of evaluation, and (3) take an advantage of Nesterov accelerated gradient to lower the required number of iterations. Eventually, we would like to increase the number of iterations by applying the techniques of bootstrapping. It is a interesting problem to extend our idea to more general machine learning algorithms such as deep neural network.

References

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. To appear in ASIACRYPT'17, 2017.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/k2). Number 2, pages 372–376, 1983.