Logistic Regression on Homomorphic Encrypted Data at Scale

Kyoohyung Han
Seoul National University
Seoul, South Korea
satanigh@snu.ac.kr

Seungwan Hong
Seoul National University
Seoul, South Korea
swanhong@snu.ac.kr

Jung Hee Cheon
Seoul National University
Seoul, South Korea
jhcheon@snu.ac.kr

Daejun Park
University of Illinois
Urbana-Champaign, IL, USA
dpark69@illinois.edu

Abstract

Machine learning on (homomorphic) encrypted data is a cryptographic method for analyzing private and/or sensitive data while keeping privacy. In the training phase, it takes as input an encrypted training data and outputs an encrypted model without ever decrypting. In the prediction phase, it uses the encrypted model to predict results on new encrypted data. In each phase, no decryption key is needed, and thus the data privacy is ultimately guaranteed. It has many applications in various areas such as finance, education, genomics, and medical field that have sensitive private data. While several studies have been reported on the prediction phase, few studies have been conducted on the *training* phase.

In this paper, we present an efficient algorithm for logistic regression on homomorphic encrypted data, and evaluate our algorithm on real financial data consisting of 422,108 samples over 200 features. Our experiment shows that an encrypted model with a sufficient Kolmogorov Smirnow statistic value can be obtained in $\sim$17 hours in a single machine. We also evaluate our algorithm on the public MNIST dataset, and it takes $\sim$2 hours to learn an encrypted model with 96.4% accuracy. Considering the inefficiency of homomorphic encryption, our result is encouraging and demonstrates the practical feasibility of the logistic regression training on large encrypted data, for the first time to the best of our knowledge.

1 Introduction

Suppose multiple financial institutions want to predict the credit scores of their customers. Although each institution could independently learn a prediction model using various machine learning techniques, they may be able to collectively learn a better model by considering all of their data together for training. However, it is risky in terms of data security to share financial data between institutions, being even illegal in many countries.

Homomorphic encryption (HE), an encryption scheme that allows arbitrary computations on encrypted data,[1] can be used to solve this dilemma. Using HE, multiple institutions can share their data in an encrypted form and run machine learning algorithms on the encrypted data without ever decrypting. Figure 1 illustrates this idea. First, each data owner shares the public key of the HE scheme and uploads his data to the cloud after encrypting with HE. Then the cloud performs the machine learning training on the encrypted data and outputs an encrypted model. This model can be decrypted by a decryptor who owns the private key. Here the decryptor can be either a single entity or a group of entities that have their own share of the private key.[2] Note that, in this HE-based approach, no information is revealed to each other except the learned model unless the underlying HE scheme is broken or its secret key is disclosed. This is the case even if the cloud is compromised. Thus it could be an ultimate solution for analyzing private or sensitive data while keeping privacy.

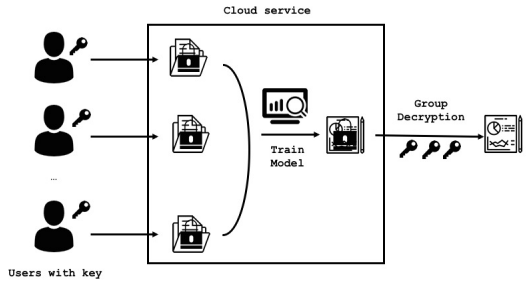

Figure 1: Machine learning on homomorphic encrypted data

This HE-based approach is also flexible in that the training computation can be *delegated* to any party (or even an *untrusted* third party) without revealing the training data (other than their own). This flexibility is desirable, as other approaches require additional assumptions and conditions that may not be realizable in practice. For example, in the multi-party computation (MPC)-based approaches (see Section 5), it is not straightforward to delegate the machine learning training to untrusted (or colluding) third parties. Also, they incur large communication overhead, where the

[1] Precisely, it is a *fully* homomorphic encryption (FHE) that supports the *unlimited* number of operations on ciphertexts. However, throughout the paper, we will refer to it as simply HE as long as the precise meaning is clear in the context.

[2] In the latter case, an additional key sharing protocol is required for the public key to be jointly generated from the entities that have a random share of the private key, by using, e.g., the threshold HE schemes (Jain, Rasmussen, and Sahai 2017; Cramer, Damgård, and Nielsen 2001).

communication overhead increases drastically as the number of participants increases. Moreover, they require all of the participants to be online during the entire training process, which adds another limitation in practice.

Despite many advantages, however, HE has not been used for computation-intensive tasks such as machine learning (especially on the training phase), having been thought to be impractical due to its large computation overhead. Indeed, basic operations (e.g., addition or multiplication) on ciphertexts are several (i.e., three to seven) orders of magnitude slower than the corresponding operations on plaintexts even in the state-of-the-art (Brakerski 2012; Brakerski, Gentry, and Vaikuntanathan 2012; Cheon et al. 2017).

In addition to the sheer amount of computation, the use of various complex operations, such as floating-point arithmetic and non-polynomial functions (e.g., sigmoid), makes it challenging to apply HE to machine learning algorithms. Indeed, HEs have been applied to machine learning algorithms only in non-realistic settings (Graepel, Lauter, and Naehrig 2012; Kim et al. 2018a) where only small-size training datasets over a small number of features are considered; or, they have been applied only on the prediction phase (Gilad-Bachrach et al. 2016; Li et al. 2017; Juvekar, Vaikuntanathan, and Chandrakasan 2018; Bourse et al. 2017) where the amount of computation is much smaller than that of the training phase.

Contributions In this paper, we present an efficient algorithm for logistic regression on homomorphic encrypted data, and demonstrate its practical feasibility against realistic size datasets, for the first time to the best of our knowledge. We evaluate our algorithm against a real, private financial dataset consisting of 422,108 samples over 200 features. Our implementation successfully learned a quality model in $\sim$17 hours on a single machine, where we tested it against a validation set of 844,217 samples and obtained a sufficient Kolmogorov Smirnow statistic value of 50.84. The performance is "only" two to three orders of magnitude slower than that of plaintext learning, which is encouraging, considering the inherent computational overhead of HEs. We also executed our algorithm on the public MNIST dataset for more detailed evaluation, and it took $\sim$2 hours to learn an encrypted model with 96.4% accuracy.

The principal techniques used to achieve this result are two-fold. First, we adopt the *approximate* HE scheme and the *approximate* bootstrapping method to reduce the computational overhead. The approximate HE can quickly compute approximated results of complex operations, avoiding the bit-manipulation overhead. (Refer to Section 5 for comparison with other HE schemes.) Similarly, the approximate bootstrapping can efficiently bootstrap a ciphertext at the cost of additional approximation noise.[3]

Also, we carefully design the logistic regression algorithm to be suitable for the HE scheme, and further optimize it to improve performance and reduce the approximation noise. Specifically, we vectorize our logistic regression algorithm to utilize the HE-specific, single-instruction-multiple-data (SIMD) operations. Also, we parallelize the bootstrapping, the most expensive HE operation, by splitting a ciphertext, while we carefully design the partition of a training dataset to avoid reconstructing the split ciphertexts, which significantly reduces the parallelization overhead. We also fine-tune the evaluation order to minimize the accumulated approximation noises due to the approximate HE scheme.

[3]We empirically show that the approximation noise is not significant to deteriorate the overall learning performance. Refer to Section 4 for more details.

2 Preliminaries

Here we provide background on homomorphic encryption.

Fully Homomorphic Encryption Fully homomorphic encryption (FHE) is an encryption scheme that allows arbitrary computation on ciphertexts without decrypting. The first secure FHE (based on the hardness assumption of a plausible number-theoretic problem) was proposed by Gentry (Gentry 2009). He first constructed a scheme, so-called somewhat homomorphic encryption (SHE), that allows a limited number of addition and multiplication operations.[4] A notable aspect of the scheme is the addition of a random noise for each encryption. The SHE scheme allows only a limited number of operations, since the plaintext may not be recovered once more than the limited number of operations has been performed, due to the noise accumulated in the ciphertext by the operations. To address the limitation, he proposed the so-called *bootstrapping* procedure that converts a ciphertext with large noise into another ciphertext with the same plaintext but small noise. Using the bootstrapping, he constructed an FHE scheme on top of the SHE scheme.

Various FHE schemes have been proposed since Gentry's construction. Their message space is either $\mathbb{Z}_p$ or a vector space over $\mathbb{Z}_p$. In a bit-wise FHE ($p = 2$), bit-manipulation and bootstrapping are efficient, but integer arithmetic is not. In a word-wise FHE ($p \gg 2$), however, the integer arithmetic is efficient as long as the result is smaller than p.

Recently, an approximate FHE scheme has been proposed by (Cheon et al. 2017). The scheme, called HEAAN, supports efficient approximate computation. In addition to addition and multiplication, it supports a rounding operation, called *rescaling*, that is essential for approximate real arithmetic (e.g., floating-point arithmetic).

HEAAN Scheme In HEAAN, a vector of real numbers is encrypted in a single ciphertext. Below are the operations (over ciphertext) provided by HEAAN.

- **Addition/Multiplication**: Point-wise addition and multiplication over (encrypted) vectors.
- **Rotation**: Given a ciphertext c of a plaintext vector $(m_1, \cdots, m_n)$, the rotation of c with k returns a new ciphertext c' that encrypts the rotated vector $(m_{k+1}, \cdots, m_n, m_1, \cdots, m_{k-1})$.
- **Rescaling**: Given a ciphertext c of a plaintext vector $(m_1, \cdots, m_n)$, the rescaling of c with k returns a new ciphertext c' that encrypts the rescaled (i.e., rounded) vector $(m'_1, \cdots, m'_n)$ where $m'_i = \lfloor m_i \cdot 2^{-k} \rceil$ for $1 \leq i \leq n$.

[4]Note that an arbitrary computation can be composed of addition and multiplication on $\mathbb{Z}_2$.

- **Bootstrapping**: Converting a ciphertext with large noise to another ciphertext of the same plaintext with small noise.

The computation cost of each operation is different. The cost comparison is roughly as follows:[5] rescaling (10ms) $<$ addition (50ms) $<$ rotation (1s) $<$ multiplication (2s) $\ll$ bootstrapping ($\sim$10 mins).

3 Logistic Regression on Encrypted Data

We present our algorithm for efficient logistic regression on homomorphic encrypted data. We first explain a baseline (plaintext) logistic regression algorithm, designed to be friendly to homomorphic evaluation (Section 3.1). Then we explain our optimization of the baseline algorithm for efficient homomorphic evaluation (Sections 3.2). Here, due to the space limit, we omit the detailed algorithm, referring the readers to the companion technical report (Han et al. 2018).

3.1 HE-Friendly Logistic Regression Algorithm

We design the baseline algorithm to be friendly to homomorphic evaluation by avoiding the use of certain types of computations that are expensive in HEs.

Mini-Batch Gradient Descent We use the mini-batch gradient descent method, where we set the mini-batch size based on the number of slots in a packed ciphertext, so as to maximize the utilization of the packed ciphertext capacity.

Nesterov Accelerated Gradient Optimizer We adopt Nesterov accelerated gradient (NAG) as the gradient descent optimization method. We choose NAG among the various optimization methods, since it provides decent optimization performance without using the division operation that is expensive in HEs.

Polynomial Approximation of Activation Function An essential step of the logistic regression training is to apply an activation function, e.g., the sigmoid function $\sigma(x) = 1/(1+e^{-x})$. Since non-polynomials are very expensive to evaluate in HEs, we consider its (low-degree) polynomial approximation σ' as an alternative in our algorithm. We use the least squares fitting method (Kreyszig 2011) to approximate the sigmoid function. The least squares fitting polynomial provides a sufficient approximation within the given interval. Figure 2, for example, plots the original sigmoid function, its least squares fitting polynomial (of degree 3) within the interval $[-8, 8]$, and its Taylor expansion (of degree 3) at the point $x = 0$. Note that the Taylor polynomial provides an accurate approximation only around the given point, while the least squares fitting polynomial provides a good approximation in a wider range.

3.2 HE-Optimized Logistic Regression Algorithm

Now we optimize the baseline algorithm to be efficiently evaluated in HEs against large encrypted data. Conceptually, the optimization is two-fold: vectorization using homomorphic SIMD operations, and fine-tuning the evaluation order.

[5]The time is measured in the machine specified in Section 4.

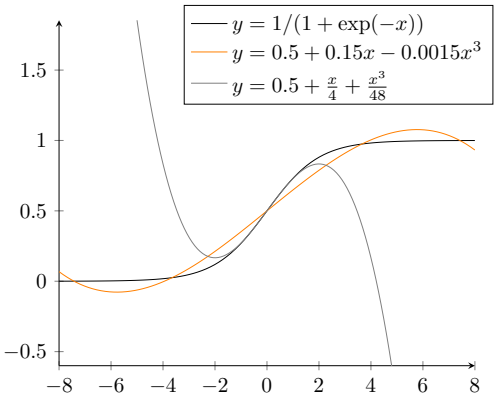

Figure 2: Sigmoid (the first) and its two approximations using the least squares fitting method (the second) and the Taylor expansion (the third).

HE-Specific Vectorization of Logistic Regression The approximate HE scheme we use supports the packing method (Cheon et al. 2017) which can further reduce the computation overhead. In the packed HEs, a single ciphertext represents an encryption of a vector of plaintexts, and ciphertext operations correspond to point-wise operations on plaintext vectors, i.e., single-instruction-multiple-data (SIMD) operations.

To maximize the benefits of the packed scheme, we vectorize our logistic regression algorithm to utilize the SIMD operations as much as possible. For example, multiple inner products can be performed by a SIMD-multiplication followed by several rotations and SIMD-additions (Han et al. 2018). Moreover, we carefully design the vectorization to minimize redundant computations caused by the use of the SIMD operations, reduce the depth of nested multiplications, and minimize the approximation noises by reordering operations. Refer to (Han et al. 2018) for more details.

Parallelized Bootstrapping One of the most expensive operations of HEs is the bootstrapping operation (even with the approximate bootstrapping method). This operation needs to be periodically executed during the entire computation. In logistic regression, for example, it should be executed every few iterations, and dominates the overall training time. It is critical for performance to optimize the bootstrapping operation.

We design our algorithm to parallelize the bootstrapping operation. It splits a ciphertext into multiple smaller chunks and executes bootstrapping on each chunk in parallel, achieving a significant speedup of the overall performance. Moreover, we carefully design the packing of training data (see below) so that our algorithm continues to use the chunks without merging them in the next training iterations, which additionally saves time it takes to reconstruct a ciphertext from the chucks.

HE-Optimized, Efficient Partition of Training Data As mentioned above, we pack multiple plaintexts in a single ciphertext, and it is critical for performance how to pack (i.e., partition) the training dataset. The training data can be seen

as an $n \times m$ matrix with n samples and m features. A naive encoding would pack each row (or column) into a ciphertext, resulting in a total of n (or m) ciphertexts. This encoding, however, is not efficient, since it either does not utilize the maximum capacity of the ciphertexts, or requires too much capacity, increasing the computation overhead drastically.

We design an efficient partition of training data in which a sub $n' \times m'$ matrix is packed into a single ciphertext, where the size of the matrix is set to the maximum capacity of each ciphertext, and m' is set to align with the aforementioned parallelization technique, avoiding an extra overhead of the ciphertext reconstruction.

4 Evaluation

We evaluate our algorithm of logistic regression on homomorphic encrypted data using both a real financial training dataset and the MNIST dataset. Our artifact is publicly available at (Han 2018).

Real Financial Dataset We executed our algorithm on a private, real financial dataset to evaluate the efficiency and the scalability of our algorithm on a large dataset.

The *encrypted* dataset we consider to evaluate our logistic regression algorithm is the real consumer credit information maintained by a credit reporting agency, Korea Credit Bureau (KCB). Owned by nine major financial institutions in Korea, KCB provides and analyzes the credit information on individuals. The dataset (for both training and validation), randomly sampled by KCB, consists of 1,266,325 individuals' credit information over 200 features that are used for credit rating. Examples of the features are the loan information (such as the number of credit loans and personal mortgages), the credit card information (such as the average amount of credit card purchases and cash advances in the last three months), and the delinquency information (such as the days of credit card delinquency). The samples are labeled with a binary classification that refers to whether each individual's credit rating is below the threshold.

We executed our logistic regression algorithm on the encrypted training set of 422,108 samples over 200 features. Having 200 iterations, it took 1,060 minutes to learn an *encrypted* model, i.e., ∼5 minutes per iteration on average, in a machine with IBM POWER8 (8 cores, 4.0GHz) and 256GB RAM. We sent the learned model to the data owner, KCB, and they decrypted and evaluated it on the validation set of 844,217 samples, having 80% accuracy and the KS value of 50.84. KCB confirmed that it provides a sufficient accuracy compared to their internal model learned using the plaintext dataset.[6] They also confirmed that our learned model gives appropriate weights on the important features (e.g., delinquency, loan, and credit card information) as expected.

Table 1 shows the detailed result of our experiment. We set the learning rate to 0.01, and the mini-batch size to 512. The ciphertext size of each mini-batch block is 4.87 MB, and thus the total size of the encrypted dataset is ∼4 GB = 4.87 MB × (422,108 / 512). The public key size is ∼2 GB.

[6]According to their report, it took several minutes to learn a model on the plaintext using the same algorithm, and the model provides the KS value of 51.99.

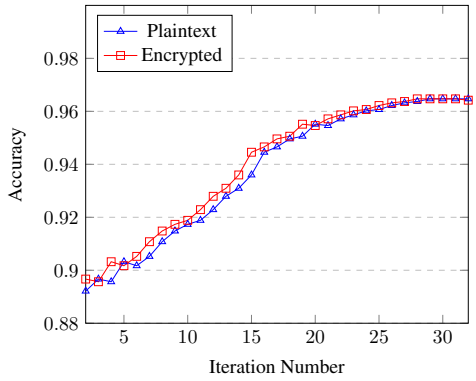

Figure 3: Comparison between encrypted and plaintext training

MNIST We executed our logistic regression algorithm on the public MNIST dataset for more detailed evaluation.

We took the MNIST dataset (LeCun, Cortes, and Burges 1999), and restructured it for the binary classification problem between 3 and 8. We compressed the original images of 28×28 pixels into 14×14 pixels, by compressing 2×2 pixels to their arithmetic mean. The restructured dataset consists of 11,982 samples of the training dataset and 1,984 samples of the validation dataset.

We encrypted the MNIST dataset and executed our logistic regression algorithm. Table 1 shows the result. With 32 iterations, our logistic algorithm took 132 minutes to learn an encrypted model. The average time for each iteration is ∼4 minutes, which is similar to that of the financial dataset, as expected. We decrypted the learned model and evaluated it on the validation dataset, obtaining 96.4% accuracy.[7]

Microbenchmarks We also executed our logistic regression algorithm on the plaintext dataset, and compared the result to that of the ciphertext learning. Recall that the approximate HE used in our algorithm introduces the approximation noise for each computation step, but it had not been clear how much the noise affects the overall training process. To evaluate the impact of the approximation noise on the overall learning performance (e.g., the convergence rate and accuracy), we measured the accuracy for each iteration for both plaintext and ciphertext training, and compared those results. Figure 3 shows the comparison result. It shows that the accuracy for each iteration in the ciphertext training is marginally different from that of the plaintext, especially in the early stage of the training process, but they eventually converged at the final step. This result implies that the additional noise introduced by the approximate HE evaluation is not significant to deteriorate the accuracy of a learned model and the training performance.

For more detailed evaluation and discussion, we refer the readers to the companion technical report (Han et al. 2018).

[7]The accuracy seems to be lower than the usual, but the difference is mainly due to the image compression. See the microbenchmark result.

Data			Logistic Regression		Accuracy		Memory		Running Time	
Financial	# Samples (training)	422,108	# Iterations	200	Accuracy	80%	Public Key	Encrypted Block	Total	Time / Iter.
	# Samples (validation)	844,217	Learning Rate	0.01	AUROC	0.8	≈ 2 GB	4.87 MB	1060 min	5.3 min
	# Features	200	Mini-batch Block Size	512	K-S value	50.84				
MNIST	# Samples (training)	11,982	# Iterations	32	Accuracy	96.4%	Public Key	Encrypted Block	Total	Time / Iter.
	# Samples (validation)	1,984	Learning Rate	1.0	AUROC	0.99	≈ 1.5 GB	3.96 MB	132 min	4.1 min
	# Features	196	Mini-batch Block Size	1024	K-S value	N/A				

Table 1: Result of machine learning on encrypted data

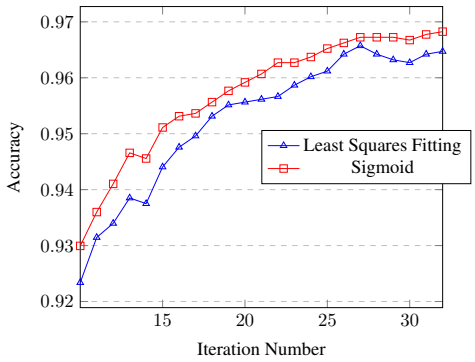

Figure 4: Comparison between sigmoid and least squares fitting (of degree 3)

Discussion It is not straightforward to provide the fair comparison of our performance with those of the related works, since the previous HE-based approaches are *not* capable of admitting such realistic size training data considered in this paper, and the MPC-based approach can be used in a limited environment where either the number of participants is small, or the two servers are trusted to not collude. As a rough comparison, however, the recent MPC-based approach (Mohassel and Zhang 2017) will take minutes[8] to learn a model on the MNIST dataset used in this paper, which is one or two orders of magnitude faster than ours.

Our algorithm requires the number of iterations to be provided in advance, which is inevitable due to the security of the underlying HE schemes. In our experiment on the financial data, the number was obtained by asking the data owner to provide a rough bound. We note that, however, one can use our algorithm in an interactive way that the data owners decrypt the learned model periodically (e.g., every 100 iterations), and decide whether to proceed further or not, depending on the quality of the model at the moment.

5 Related Work

There have been several studies on performing a machine learning without revealing private information. Here we consider two major types of approaches: HE-based approaches and the multi-party computation (MPC)-based approaches.

[8]The time is obtained by extrapolating their experimental result on the MNIST dataset.

HE-Based Approaches Graepel *et al*. (Graepel, Lauter, and Naehrig 2012) presented a homomorphic evaluation algorithm of two binary classifiers (i.e., linear means and Fisher's linear discriminant classifiers), and Kim *et al*. (Kim et al. 2018b; 2018a) proposed a homomorphic evaluation of logistic regression. However, they provided only a proof-of-concept evaluation, where small-scale training datasets (consisting of only dozens of samples and features) are considered. Moreover, it is not clear how scalable their approaches are as the size of datasets and the number of iterations increase. Indeed, their implementations require the multiplication depth (i.e., the number of iterations) to be bounded, meaning that their implementations are not scalable. Our algorithm, however, is scalable in the sense that it can admit an *arbitrary* number of iterations, and the time complexity is *linear* in terms of the number of iterations.

There also have been reported studies on homomorphic evaluation of the prediction phase of machine learning algorithms including neural networks (Gilad-Bachrach et al. 2016; Li et al. 2017; Juvekar, Vaikuntanathan, and Chandrakasan 2018; Bourse et al. 2017). However, the prediction phase is much simpler than the training phase in terms of the amount of computation (especially in terms of the multiplication depth), and thus their techniques are hard to be applied to the training phase directly.

On the other hand, Aono *et al*. (Aono et al. 2016) presented a protocol for secure logistic regression using the additively homomorphic encryption. It approximates the cost function by a low-degree polynomial, and encrypts the training data in the form of the monomials of the polynomial. Then, the approximated cost function can be homomorphically evaluated by simply adding the encrypted monomials. This protocol, however, has the disadvantage that the number and/or the size of ciphertexts increase exponentially as the degree of the polynomial approximation increases.

MPC-Based Approaches Nikolaenko *et al*. (Nikolaenko et al. 2013) proposed an MPC-based protocol for training linear regression model, which combines a linear homomorphic encryption and the Yao's garbled circuit construction (Yao 1986). Mohassel and Zhang (Mohassel and Zhang 2017) improved the protocol by using secure arithmetic operations on shared decimal numbers, and applied it to logistic regression and neural network training.

However, as mentioned eariler, the MPC-based approaches incur large communication overhead, and require

all of the participants to be online. To mitigate this problem, an approach using two delegating servers was proposed (Mohassel and Zhang 2017), where multiple parties upload their data to two servers and delegate the training task to the servers using the two-party computation (2PC). This approach, however, requires an additional assumption that two servers do not collude. Recall that our HE-based approach requires *no* assumption on the server, and can admit even a compromised server.

Other HE Schemes The bit-wise HE schemes (Ducas and Micciancio 2015; Chillotti et al. 2017) provide an efficient bootstrapping operation, and can admit a boolean circuit directly as a complex operation that involves bit-manipulation. However, their operations are inherently slow due to their large circuit depth. On the other hand, the word-wise HE schemes (Brakerski, Gentry, and Vaikuntanathan 2012; Fan and Vercauteren 2012; Brakerski 2012; Cheon et al. 2017) provide more efficient operations since their circuit depth can be significantly reduced. However, they suffer from an expensive bootstrapping operation due to the large size of ciphertexts. Also, they do not provide the same level of efficiency for complex operations that involve bit-manipulation, as their circuit depth is still large in the form of an arithmetic circuit over words.

The word-wise approximate HE scheme, adopted in our algorithm, can improve the efficiency of bit-manipulating complex operations at the cost of approximation noises. It is useful in applications where the small approximation noises in the intermediate computation steps are not critical for the final computation result, which is indeed the case for most of the machine learning algorithms.

6 Conclusion and Further Work

In this paper, we presented an efficient logistic regression algorithm on large (fully) homomorphically encrypted data, and evaluated it against both the private financial data and the public MNIST dataset. Our implementation successfully learned a quality model in about 17 and 2 hours, respectively, which demonstrates the practical feasibility of our algorithm on realistic size data. We believe that the techniques we developed here can be also readily used for homomorphically evaluating other machine learning algorithms such as neural networks, which we leave as a future work.

References

Aono, Y.; Hayashi, T.; Phong, L. T.; and Wang, L. 2016. Scalable and secure logistic regression via homomorphic encryption. Cryptology ePrint Archive, Report 2016/111.

Bourse, F.; Minelli, M.; Minihold, M.; and Paillier, P. 2017. Fast homomorphic evaluation of deep discretized neural networks. Cryptology ePrint Archive, Report 2017/1114.

Brakerski, Z.; Gentry, C.; and Vaikuntanathan, V. 2012. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 309–325. ACM.

Brakerski, Z. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, 868–886.

Cheon, J. H.; Kim, A.; Kim, M.; and Song, Y. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, 409–437. Springer.

Chillotti, I.; Gama, N.; Georgieva, M.; and Izabachène, M. 2017. Improving TFHE: faster packed homomorphic operations and efficient circuit bootstrapping.

Cramer, R.; Damgård, I.; and Nielsen, J. B. 2001. Multiparty computation from threshold homomorphic encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 280–300. Springer.

Ducas, L., and Micciancio, D. 2015. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology–EUROCRYPT 2015*. Springer. 617–640.

Fan, J., and Vercauteren, F. 2012. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144.

Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, 169–178.

Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; and Wernsing, J. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, 201–210.

Graepel, T.; Lauter, K.; and Naehrig, M. 2012. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, 1–21. Springer.

Han, K.; Hong, S.; Cheon, J. H.; and Park, D. 2018. Efficient logistic regression on large encrypted data. Cryptology ePrint Archive, Report 2018/662.

Han, K. 2018. `https://github.com/HanKyoohyung/Logistic_Regression_on_Encrypted_Data`.

Jain, A.; Rasmussen, P. M.; and Sahai, A. 2017. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive* 2017:257.

Juvekar, C.; Vaikuntanathan, V.; and Chandrakasan, A. 2018. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association.

Kim, A.; Song, Y.; Kim, M.; Lee, K.; and Cheon, J. H. 2018a. Logistic regression model training based on the approximate homomorphic encryption.

Kim, M.; Song, Y.; Wang, S.; Xia, Y.; and Jiang, X. 2018b. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics* 6(2):e19.

Kreyszig, E. 2011. *Advanced Engineering Mathematics*. Wiley, 10th edition.

LeCun, Y.; Cortes, C.; and Burges, C. J. 1999. The MNIST Database of Handwritten Digits.

Li, P.; Li, J.; Huang, Z.; Gao, C.-Z.; Chen, W.-B.; and Chen, K. 2017. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing* 1–10.

Mohassel, P., and Zhang, Y. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 19–38. IEEE.

Nikolaenko, V.; Weinsberg, U.; Ioannidis, S.; Joye, M.; Boneh, D.; and Taft, N. 2013. Privacy-preserving ridge regression on hundreds of millions of records. In *Security and Privacy (SP), 2013 IEEE Symposium on*, 334–348. IEEE.

Yao, A. C.-C. 1986. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, 162–167. IEEE.