Efficient Homomorphic Evaluation on Large Interval

Jung Hee Cheon,[1],[2] Wootae Kim,[1], and Jai Hyun Park(✉)[1]

[1] Department of Mathematical Sciences, Seoul National University, South Korea
{jhcheon, wo01416, jhyunp}@snu.ac.kr
[2] Crypto Lab Inc., South Korea

Abstract. Homomorphic encryption (HE) is being widely used for privacy-preserving computation. Since HE schemes only support polynomial operations, it is prevalent to use polynomial approximations of non-polynomial functions. We cannot monitor the intermediate values during the evaluation; as a consequence, we should utilize polynomial approximations with sufficiently large approximation intervals to prevent the failure of the evaluation. However, the large approximation interval potentially accompanies computational overhead, and it is a serious bottleneck of HE application on real data.

In this work, we introduce domain extension polynomials (DEPs) that extend the domain interval of functions by a factor of k while preserving the feature of the original function on its original domain interval. By repeatedly iterating the domain-extension process with DEPs, we can extend with $O(\log K)$ multiplications the domain of given function by a factor of K while the feature of the original function is preserved on its original domain interval.

By using DEPs, we can efficiently evaluate in encrypted state a function that converges at infinities. To uniformly approximate the function on $[-R, R]$, our method exploits $O(\log R)$ multiplications and $O(1)$ memory. This is more efficient than the current best approach, the minimax approximation and Paterson-Stockmeyer algorithm, which uses $O(\sqrt{R})$ multiplications and $O(\sqrt{R})$ memory for the evaluation. As another application of DEPs, we also suggest a method to manage the risky outliers from a wide interval $[-R, R]$ by using $O(\log R)$ additional multiplications.

As a real-world application, we exploit our uniform approximation of the logistic function on wide intervals to logistic regression. We trained the model on large public datasets in encrypted state using the polynomial approximation of the logistic function on $[-7683, 7683]$.

1 Introduction

There have been many attempts for privacy-preserving delegation of computation these days. One of the most popular solutions for the privacy-preserving computation is homomorphic encryption (HE) that supports several operations

between ciphertexts without any decryption process. The privacy-preserving delegation of machine learning (ML) algorithms based on HE, in particular, has garnered much attention since many ML algorithms are being utilized for personal data, and privacy-related issues are constantly being raised.

However, HE serves only a limited types of operations: addition and multiplication. It has been a challenging problem to compute a general circuit containing sophisticated non-polynomial functions based on HE. The most prevalent solution for this is to replace non-polynomial functions by polynomials. An evaluator of HE first selects the domain of each real-valued function, and replaces it by its polynomial approximation on the selected domain interval. This enables us to approximately compute any given circuit in encrypted states, and now we focus on a specific problem that is to find an appropriate polynomial approximation for HE.

Meanwhile, a computation over encrypted data has several different points compared to that n unencrypted state. First, the evaluator of HE cannot observe the input values, and moreover, the intermediate values during the evaluation should not be revealed. As a consequence, there are only a few clues about the domain interval of each inner function. Also, all intermediate values during the evaluation should not blow up. HE restricts us from monitoring the intermediate values. Once a single intermediate value gets out of the plaintext space of HE, it has a potential to ruin the ciphertext and damage all evaluation results. For these reasons, we should find polynomial approximations on large domain intervals for the successful evaluation based on HE.

1.1 Fundamental Problem

The polynomial approximation accompanies some errors compared to the original function, and the error determines the quality of computation. To use a polynomial approximation for the privacy-preserving computation based on HE, we should consider both (a) the error induced by the polynomial approximation and (b) the computational cost for the evaluation of the polynomial approximation function. Ultimately, the approximate computation with a small error and a small computational cost is most desirable. For the ease of discussion, in this paper, we aim to minimize the computational cost to (approximately) evaluate a real-valued function in encrypted state under a fixed maximum error.

For the computational cost, we take the number of multiplications into account as a top priority. This is because in HE computation, the multiplication between two ciphertexts is much heavier than the other homomorphic operations such as addition, subtraction, and constant multiplication. We stress out that the evaluation of a polynomial with a higher degree does not necessarily require more number of multiplications.[3] Our goal here is to minimize the computational costs during the approximate computation; rather than to minimize the degree of a polynomial approximation.

[3] For instance, we can compute a polynomial $x^8 = ((x^2)^2)^2$ with 3 multiplications while we should use at least 4 multiplications to compute x^7.

To put it all together, the substantial question is how to evaluate a non-polynomial real-valued function over homomorphically encrypted data with a less number of multiplications. However, as we point out, HE evaluation requires a sufficiently large domain interval to guarantee the success of the evaluation. While there have been some general solutions for this question, they accompany too much computational overhead as we select the domain intervals large enough. In this paper, we focus on the following question:

On a large domain interval, how to approximately evaluate a non-polynomial real-valued function over homomorphically encrypted data with a less number of multiplications?

We provide a partial solution to the above question for the large domain interval. We start with the logistic function that is widely being used for ML algorithms. While the input domain should be sufficiently large for many applications of the logistic function, the HE evaluation of the logistic function on a wide approximation interval with a small maximum error entails a considerable computational cost. With the current best approach, such as the combination of minimax approximation and Paterson-Stockmeyer algorithm, we should use $\Omega(\sqrt{R})$ multiplications for the approximate evaluation with fixed maximum error on the domain interval $[-R, R]$. We propose a better solution, *domain-extension methodology*, which enables to use $O(\log R)$ multiplications to approximately evaluate the logistic function on domain interval $[-R, R]$ with a fixed maximum error in encrypted states.

More generally, our domain-extension methodology serves an efficient privacy-preserving evaluation of functions that converge at infinities (i.e. the limit of $\sigma(x)$ exists as $x \to \pm\infty$). By using our methodology, we can approximately evaluate with a fixed maximum error those functions on $[-R, R]$ with $O(\sqrt{R})$ multiplications in encrypted states. In the best of our knowledge, this is computationally better than the combination of minimax approximation and Paterson-Stockmeyer algorithm, the current best approach, which use $\Omega(\sqrt{R})$ multiplications.

In addition, our methodology can be utilized for the management of outliers from the outside of the approximation interval during the privacy-preserving computation. Even a few outlying data can destroy the overall result because the evaluation value of polynomials can easily escape the plaintext space of HE at outside a fixed domain interval. However, selecting a huge approximation interval to contain all possible outliers accompanies too much computational overheads and makes the computation impractical. By exploiting our methodology, we can accommodate the rare outliers from a sufficiently big interval $[-R, R]$ by using $O(\log R)$ additional multiplications.

1.2 Our Contributions

Our contributions can be formalized as follows.

- **Domain-extension methodology** We propose domain-extension methodology, which efficiently extend the valid domain interval of a polynomial for HE. We first suggest the concept of *domain extension functions* (DEFs) and *domain extension polynomials* (DEPs), which extend the valid domain interval by a factor of L, the extension ratio (i.e., extend the domain interval from $[-R, R]$ to $[-LR, LR]$). In contrast to a simple scaling, DEFs and DEPs preserve the feature of the original function on its original domain interval. Domain-extension methodology is to use DEFs and DEPs repeatedly. After n iterations of domain-extension process using DEFs and DEPs, we can extend the domain interval by a factor of L^n while preserving the feature of the original function on its original domain interval.As a consequence, once we are given a function on a small interval $[-r, r]$, we can utilize it on a wider domain interval $[-R, R]$ by using $O(\log R)$ additional operations; while the extended function behaves similarly on the original domain interval $[-r, r]$.

- **Uniform approximation on wide intervals**. By using domain-extension methodology, we suggest an efficient way to evaluate in encrypted state a function $f(\cdot)$ that converges at infinities. *Our method provides a uniform polynomial approximation of $f(\cdot)$ on $[-R, R]$ that can be evaluated by $O(\log R)$ multiplications and $O(1)$ memory in encrypted state.*
  In terms of computational complexity, this is much better than the current best approach, minimax approximation and Paterson-Stockmeyer algorithm. We observe that the degree of minimax approximation on $[-R, R]$ is $\Omega(R)$ in cases of functions that converges at infinities. As a result, during the computation in encrypted states, it accompanies a heavy computational overhead, $\Omega(\sqrt{R})$ multiplications and $\Omega(\sqrt{R})$ space cost, and it is impractical to use it for sufficiently big R. In contrast, our method can be evaluated with $O(\log R)$ operations, and is more practical. We also implement and compare both methods in Section 5.1.

- **Accommodation of outliers from wide intervals**. While using HE for numerous data, a single unexpected datum has a potential to damage the overall result. This is due to the fact that an evaluation of a polynomial at the out of a fixed interval can easily gets out of the plaintext space of HE. This immediately contaminate the ciphertext, and ruin the overall evaluation results. However, it seems to be impractical to use polynomial approximations on huge approximation intervals that are capable to contain all outliers.
  On the other hand, by using domain-extension method, we can efficiently accommodate the rare outliers from wide intervals. Our method accommodates the outliers from a wide interval $[-R, R]$ by using $O(\log R)$ additional operations, and prevent the rare outliers from ruining the entire process.

- **Logistic regression over large datasets in encrypted state**. We apply our methodology to the logistic function, and yield an efficient way to evaluate the logistic function on wide domain intervals in encrypted state. By using it, we implement the privacy-preserving logistic regression for real heavy datasets based on HE. There have been several similar works to this,

but in the best of our knowledge, they are unlikely to be applied to heavy datasets. All previous works heuristically selected the domain interval of the logistic function, and we observed that those short intervals are not sufficient for many of heavy public datasets. Moreover, previous works select the domain intervals by monitoring the process in unencrypted state; however, this monitoring is undesirable in HE applications since it may leak some pieces of information about the data. In this work, on the other hand, we approximate the logistic function on sufficiently large domain intervals by using DEPs. This enables us to successfully perform the logistic regression on heavy datasets with various hyper-parameters in encrypted states.

1.3 Related Work

Machine Learning over Encrypted Data For the privacy-preserving delegation of computation, many works have applied homomorphic encryption to machine learning algorithms. The key of HE-based solutions for privacy-preserving ML algorithms are appropriate replacement of real-valued functions by their polynomial approximations. Since logistic regression has a simple structure, there have been several works that leverage HE for the privacy-preserving logistic regression [1–4]. Kim et al. proposed a HE based logistic regression [2]. In order to evaluate the logistic functions based on HE, they replace the logistic function by the least square fitting polynomial approximation on $[-8, 8]$. Chen et al. implement HE based logistic regression by using the minimax polynomial approximation on $[-5, 5]$ [3]. To make input values of the logistic function belong to $[-5, 5]$, they perform preprocessing (average pooling) to the data. Han et al. use the least square fitting polynomial approximation on $[-8, 8]$, and also perform preprocessing (average pooling) for the experiment on MNIST dataset [4]. In the best of our knowledge, all previous works heuristically select the input range of the logistic function. In some cases, preprocessing is needed to force the inputs of the logistic function to belong to the selected interval. This selection of narrow approximation interval makes it difficult to perform the logistic regression on large public datasets such as Swarm Behavior dataset. In this work, we suggest a practical solution to approximate the logistic function on substantially wide intervals, and by using it, we implement the logistic regression in encrypted state for large datasets.

There also have been many works that utilized HE to ML algorithms other than the logistic regression. CryptoNet modified CNN models to be HE friendly by replacing ReLU function and max pooling by square function and sum function respectively [5]. Hesamifard et al. suggested a better polynomial replacement of ReLU function and analysed its efficiency [6].

Computation with less multiplications When we use HE, multiplication is much heavier than the other operations such as addition and scalar multiplication. Thus, the evaluation of a polynomial by using a less number of multiplications is important to mitigate the computational overhead accompanied by HE.

Paterson-Stockmeyer algorithm is an algorithm to evaluate a polynomial of degree d by using $O(\sqrt{d})$multiplications [7]. Chen et al. adopted Paterson-Stockmeyer algorithm for the HE computations, and they performed Paterson-Stockmeyer algorithm based on the Chebyshev basis to make the coefficients more consistent [8]. Chen et al.'s algorithm is being widely used for evaluation of high degree polynomials based on HE [9, 10].

On the other hand, while Paterson-Stockmeyer algorithm provides good performances on general polynomials, there exists some polynomials that can be evaluated with less number of multiplications compared to Paterson-Stockmeyer algorithm. For example, x^{2^n} can be evaluated with only n multiplications by iterative squaring, but Paterson-Stockmeyer algorithm uses about $2^{n/2+1}$ multiplications to evaluate it. Finding an approximate polynomial that can be evaluated with less number of multiplications can reduce the computational cost. Cheon et al. [11] propose an iterative method to approximately evaluate the sign function on $[-1, 1]$, and they argue that their method requires relatively less number of multiplications. In the same manner, we suggest an iterative method for the polynomial approximation on a wide domain interval, which can be evaluated with less number of multiplications.

2 Preliminaries

2.1 Notations

In this paper, $C_{\lim}$ is the class of continuous functions that converges at infinities (i.e., $C_{\lim} := \{f(\cdot) |\ f(\cdot)$ is continuous, and limits of $f(x)$ exists as $x \to \pm\infty\}$. When we say *multiplication*, it means a non-scalar multiplication unless we indicate. A maximum error between two function $f(\cdot)$ and $g(\cdot)$ on an interval I is $\|f - g\|_{\infty,I} := \sup\{|f(x) - g(x)| : x \in I\}$. We omit I unless it is necessary.

For the homomorphic encryption, we use the ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ where N is a power-of-two integer, and $\mathcal{R}_q$ denotes $\mathcal{R}/q\mathcal{R}$.

2.2 Homomorphic Encryption and CKKS scheme

Homomorphic Encryptoin (HE) is a cryptographic scheme that allows operations in encrypted states without any decryption process. HE is widely used for fully privacy-preserving delegation of computation including machine learning circuits such as logistic regression. Among various HE schemes, we adopted CKKS scheme presented by Cheon at el [12, 13], which support arithmetic operation of approximate numbers . CKKS scheme has a strong advantage on application to machine learning algorithms since its plaintexts are real numbers. As a matter of fact, CKKS scheme has been adopted to many implementations of privacy-preserving machine learning algorithms based on HE [4, 14, 15].

Let N be a power-of-two integer. There exists a field isomorphism $\tau : \mathbb{R}[X]/(X^N + 1) \to \mathbb{C}^{N/2}$ which CKKS scheme uses for encoding and decoding of messages.

CKKS scheme encrypts a (encoded) plaintext $\mathsf{m} \in \mathbb{C}^{N/2}$ into a ciphertext $\mathsf{ct} \in \mathcal{R}_Q$. For the decryption process, a secret key sk is needed; formally, the decryption process is

$$\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}) = \mathsf{m} + \mathsf{e},$$

where e is a small error vector. We refer [12] for the detail of CKKS scheme. The main operations of CKKS scheme and their properties are followings.

- $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$: For ciphertexts ct_1 and ct_2 of m_1 and m_2, output a ciphertext ct of $\mathsf{m}_1 + \mathsf{m}_2$.
- $\mathsf{Mult}(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{evk})$: For ciphertexts ct_1 and ct_2 of m_1 and m_2, output a ciphertext ct of $\mathsf{m}_1 \odot \mathsf{m}_2$, where $\odot$ indicates the component-wise multiplication between messages in $\mathbb{C}^{N/2}$.
- $\mathsf{Bootstrap}(\mathsf{ct}, \mathsf{evk})$: For ciphertexts ct of m, output a ciphertext ct' of m with a fresh noise level.

Each multiplication between ciphertexts increases the noise in ciphertexts, so it is necessary to refresh the noise level by using $\mathsf{Bootstrap}$ after each computation of a fixed number of multiplicative depth. Also, the bootstrapping and the multiplication between ciphertexts are significantly slower than other operations. Thereby, the number of multiplications and the multiplicative depth mostly determines the computational costs of circuits over CKKS ciphertexts.

To use CKKS scheme, we should set the parameters N and q_L, where N is the dimension of the ring $\mathcal{R}_{q_L}$ and q_L is the initial modulus size.

2.3 Minimax Polynomial Approximation and Paterson-Stockmeyer Algorithm

For a given continuous function $f(x)$, an interval $[a, b]$ and a positive integer d, a *minimax polynomial approximation* $p(x)$ is a polynomial of degree at most d such that minimize the *maximum error*, $\|f - p\|_{\infty,[a,b]}$. Many algorithms such as Remez algorithm and Barycentric-Remez algorithm have been suggested for finding the minimax polynomial approximation [16, 17].

Minimax approximation guarantees a good quality of approximation at each point of the approximation interval. Hence, it is plausible to be adopted to the polynomial approximation for HE since the inputs of each function are hidden to the HE evaluator.

The evaluation of a high degree polynomial requires lots of multiplication and addition operations. There are several evaluation algorithms such as Hornor's method [18] and Paterson-Stockmeyer algorithm [19] to decrease the computational costs of the evaluation. Hornor's method requires $O(d)$ multiplications, and Paterson-Stockmeyer algorithm requires $O(\sqrt{d})$ multiplications. Paterson-Stockmeyer algorithm is often being adopted when multiplication is substantially more expensive than addition operation; such as circuits over matrix and circuits over homomorphically encrypted data.

Chen et al. [8] suggest Paterson-Stockmeyer algorithm with Chebyshev basis. They pointed out that in the case of many known polynomial approximation

techniques, the coefficient with respect to the Chebyshev basis is more consistent compared to that with respect to the power basis. The modified Paterson-Stockmeyer algorithm is currently widely being used for the evaluation of high degree polynomials based on HE.

2.4 Logistic Regression

Logistic regression algorithm is a well-known ML algorithm that solves binary classification problems. Logistic regression model consists of a weight w and a bias b. As in [2, 20], for the ease of discussion, we use each datum with an additional feature with the value of 1 for the bias term. To be more detailed, we represent each datum x in the form of $z = (x, 1)$, and the logistic regression model in the form of $W = (w^T, b)^T$. Note that $w^T x + b = W^T z$ holds.

For the inference of logistic regression, a model W classifies each datum x into a class of either one of 1 or -1 by addressing the value of the following.

$$\Pr\Big(\textit{the label of } x \textit{ is } 1\Big) = \sigma(W^T z)$$

where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function.

For the training of logistic regression, we consider a cost function

$$J(W) = \frac{1}{n} \sum_{(z,y)} \log(1 + \exp(-y \cdot W^T z))$$

where each z is a datum with the class of y, and n is the number of data. By using gradient descent method to this cost function, the training process seeks the appropriate weights and bias that minimize the cost function for given training dataset. More precisely, for a given learning rate α, we update the weight and bias as follows.

$$W \leftarrow W - \alpha \nabla J(W) = W + \frac{\alpha}{n} \sum_{(z,y)} \sigma(-y \cdot W^T z) \cdot (yz)$$

3 Domain-extension Methodology

HE supports only polynomial operations: addition and multiplication. Consequently, when we compute a circuit over encrypted data based on HE, we need to replace the non-polynomials by their polynomial approximations on each of its estimated domain interval, *the approximation interval*. Once the intermediate value during the computation gets out of the plaintext space of HE, the outlier immediately ruins the ciphertext and contaminates the entire computation. To avoid this, we need to select the approximation interval of each polynomial approximation large enough. Unfortunately, a larger approximation interval requires a bigger degree of approximate polynomial, and result in a considerable computational overhead.

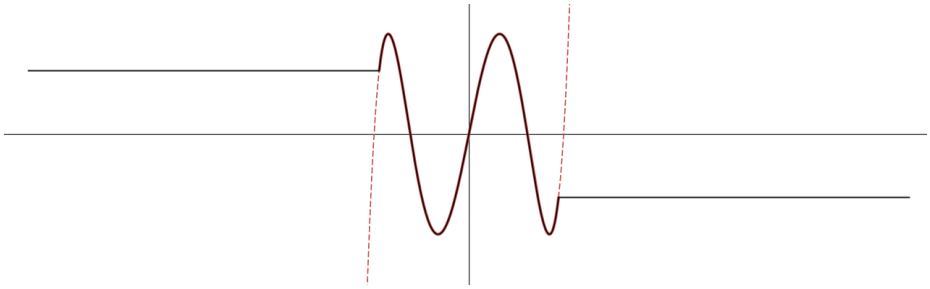


Fig. 1: An example of domain-extension process by using a DEF.

We can efficiently evaluate low-degree polynomials, but in general, a low-degree polynomial is useful only on relatively small domain intervals and behaves unexpectedly on the out of the small interval. In this work, we aim to utilize a low-degree polynomial to large intervals by additionally attaching a few number of operations. We call this process by *domain-extension* process. By the domain-extension process, we desire to use a low-degree polynomial similarly on its original (small) domain interval; meanwhile, on a larger interval, we maintain the size of function values, and prevent it from behaving unexpectedly.

For the ease of discussion, during domain-extension process, we distinguish two domain intervals of given polynomial, *the interval type I and II*. We denote the interval type I as the interval that the function after the domain-extension process behaves similar to the original polynomial. On the other hand, we aim to extend the interval type II during the domain-extension process; that is, the maximum norm (i.e., l_∞-norm) of the function after the process on the extended interval type II is bounded by that of the original function on the original interval type II. In short, we regard interval type I as where the most of inputs come from, and interval type II, on the other hand, is the interval where we want to prevent the function value from behaving unexpectedly.

3.1 Motivation

Domain Extension Function.

We first introduce *domain extension functions* (DEFs). For a given $L > 1$, we define a function $D : [-L, L] \to [-1, 1]$ by $D(x) = \frac{1}{2}\big(|x+1| - |x-1|\big)$. We will call this function a DEF with extension ratio L.

Now assume that we are given a function $P(\cdot)$ with the domain $[-1, 1]$. Then, on $[-L, L]$, the composition $P \circ D(\cdot)$ behaves as followings.

$$P \circ D(x) = \begin{cases} P(-1), & \text{if } -L \le x \le -1 \\ P(x), & \text{if } -1 \le x \le 1 \\ P(1), & \text{if } 1 \le x \le L \end{cases}$$

This means that interval type II of $P \circ D$ is extended to $[-L, L]$ while the function values does not change on the interval type I, $[-1, 1]$.

Figure 1 illustrates the feature of DEF with extension factor 5, and how it extends the small interval type II $[-1, 1]$ of a polynomial $4x^3 - 3x$ to a wide interval $[-5, 5]$. In particular, it extends interval type II by simply stretching the endpoints, and it exactly preserve the features on interval type I.

Domain-extension Methodology. We can extend the size of the interval type II exponentially by using the argument above repeatedly. To be more precise, let $D(\cdot)$ be a DEF with the extension factor L that extends interval type II $[-1, 1]$ to $[-L, L]$. We may consider the scaled function $D_n(x) := L^n D(x/L^n)$ as another DEF with extension ratio L, which extends the interval type II from $[-L^n, L^n]$ to $[-L^{n+1}, L^{n+1}]$. Applying $D_i(\cdot)$ n times sequentially, we can extend interval type II of $f(\cdot)$ from $[-1, 1]$ to $[-L^n, L^n]$. In other words, we extend the size of interval type II by factor of R with $O(\log R)$ iterations. We also point out that the function value on the original domain, $[-1, 1]$, does not change.

We name this iterative strategy that extends the interval type II while preserving the features of the target function on interval type I as *domain-extension methodology*. Unfortunately, DEFs may not be directly used for privacy-preserving computations based on HE since they cannot be evaluated by using only HE operations. Thus, we define domain extension polynomials (DEPs) in Section 3.2, and we explain the domain-extension methodology with DEPs in Section 3.3.

3.2 Domain Extension Polynomial

In order to utilize domain-extension methodology for the HE-based computations, we need polynomials that can take role of DEFs. The crux is to exploit the approximate polynomials of DEFs. For the ease of discussion, we define a narrow definition of DEPs as follows.

Definition 1 (Domain Extension Polynomials). *For given constants $R_2 > R_1 > r > 0$ and small error $\delta > 0$, we define $\mathscr{D}(\delta, r, R_1, R_2)$ as a class of polynomials $d(\cdot)$ satisfying:*

(a) $|x - d(x)| \leq \delta |x|^3 \ \forall x \in [-r, r]$
(b) $0 \leq d'(x) \leq 1 \ \forall x \in [-r, r]$
(c) $d(r) < d(x) < R_1 \ \forall x \in [r, R_2]$

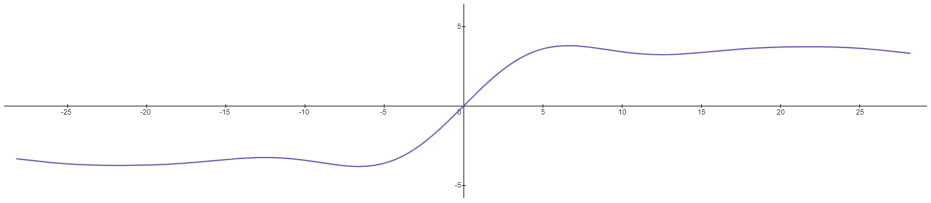


Fig. 2: An example of DEP.

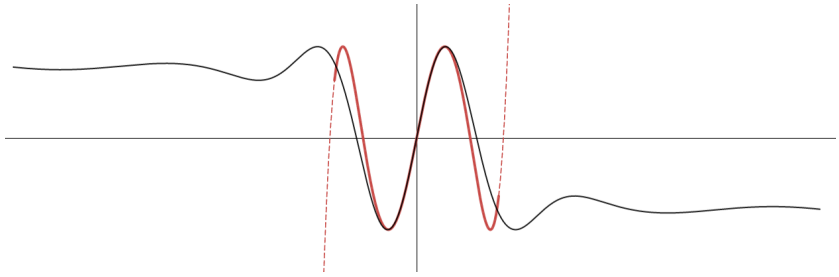


Fig. 3: An example of domain-extension process by using a DEP.

(d) $-R_1 < d(x) < d(-r)\ \forall x \in [-R_2, -r]$.
We call R_2/R_1 *as the extension ratio of* $d(\cdot)$. *We also denote* $d(\cdot)$ *is a DEP extending* $[-R_1, R_1]$ *to* $[-R_2, R_2]$ *preserving* $[-r, r]$ *if* $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$.

Figure 2 visualizes a graph of an example of DEPs. Note that it approximates a DEF. As a consequence, it may behave similar to a DEF. Figure 3 visualizes how a DEP (approximately) extend the interval type II of a given polynomial. We can observe that a DEP can extend the interval type II of a polynomial while preserving (with minor changes) the function values on a narrow interval type I. More precisely, we argue that a DEP $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$ can extend the interval type II of a polynomial from $[-R_1, R_1]$ to $[-R_2, R_2]$ while preserving the polynomial on a narrow subinterval of type I, $[-r, r]$. We formally state this in Theorem 1.

Theorem 1. *Assume that we are given a DEP* $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$, *and a polynomial* $p(\cdot)$ *such that* $\sup\{|p'(x)| : -r \le x \le r\} \le M$. *Then,*

$$\|p \circ d\|_{\infty,[-R_2,R_2]} \le \|p\|_{\infty,[-R_1,R_1]},$$

$$\|p \circ d - p\|_{\infty,[-r,r]} \le Mr^3\delta,$$

and

$$\sup\{|(p \circ d)'(x)| : -r \le x \le r\} \le M.$$

Proof. For each $x \in [-r, r]$, there exist x_* between $d(x)$ and x such that

$$|p \circ d(x) - p(x)| = |p'(x_*)||x - d(x)| \le M\delta|x|^3 \le Mr^3\delta.$$

Also,

$$|(p \circ d)'(x)| = |p'(d(x))d'(x)| \le M.$$

Note that $0 \le d' \le 1$ on $[-r, r]$ insists that $-r \le -|x| \le d(x) \le |x| \le r$. □

To put it all together, a DEP behaves similar to a DEF. A DEP extends the interval type II from $[-R_1, R_1]$ to $[-R_2, R_2]$, while preserving the features of the original function on a small subinterval of type I, $[-r, r]$.

3.3 Domain-extension Methodology with DEPs

We now describe an iterative domain-extension by using a DEP.

We begin with the observation that if $d(\cdot)$ is a DEP that extends $[-R_1, R_1]$ to $[-R_2, R_2]$, then its scaled polynomial $kd(\cdot/k)$ is a DEP that extends $[-kR_1, kR_1]$ to $[-kR_2, kR_2]$.

Theorem 2. *If $d(\cdot) \in \mathscr{D}(\delta, r, R_1, R_2)$ and $L > 1$, then*

$$D(x) := Ld\Big(\frac{x}{L}\Big) \in \mathscr{D}\Big(\frac{\delta}{L^2}, r, LR_1, LR_2\Big).$$

Proof. For each $x \in [-r, r]$, let $z = x/L \in [-r, r]$. Then,

$$|x - D(x)| = |Lz - Ld(z)| \le L\delta|z|^3 = \frac{\delta}{L^2}|x|^3$$

holds. Also,

$$D'(x) = \frac{d}{dx}\left(Ld\left(\frac{x}{L}\right)\right) = d'(x) \in [0, 1].$$

For each $x \in [r, LR_2]$, let $z = x/L \in [-r/L, R_2]$. If $z \le r$, $d\left(\frac{r}{L}\right) \le d(z) \le d(r) \le R_1$ holds since $0 \le d' \le 1$ on $x \in [0, r]$. Meanwhile, if $z > r$, $d\left(\frac{r}{L}\right) \le d(r) \le d(z) \le R_1$ holds. Thus, in both cases,

$$D(r) = Ld\left(\frac{r}{L}\right) \le Ld(z) = D(x) \le LR_1$$

holds, and similar result holds for $x \in [-LR_2, -r]$.
Therefore, $D(x) \in \mathscr{D}\Big(\frac{\delta}{L^2}, r, LR_1, LR_2\Big)$. □

By using Theorem 2, we can generate a sequence of DEPs, $B_n(x) = L^n B(x/L^n)$, from a base DEP $B(x)$ that extends $[-R, R]$ to $[-LR, LR]$. Each DEP $B_n(\cdot)$ extends $[-L^n R, L^n R]$ to $[-L^{n+1}R, L^{n+1}R]$, so we can utilize them sequentially. To apply DEPs $B_i(\cdot)$ sequentially to a given polynomial that has interval type II $[-R, R]$, the interval type II would be extended to $[-L^n R, L^n R]$ after n iterations. Hence, we can extend the interval type II by a factor of L' with $O(\log L')$ iterations. Figure 4 visualizes how the sequence of DEPs extends interval type II for each iteration. We shall call this methodology as *domain-extension methodology*. We argue that domain-extension methodology with DEPs (1) extends the interval type II efficiently, and (2) preserves the feature of the original function on interval type I.

Theorem 3 (Domain-extension methodology). *Assume that $B(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ and a polynomial $p(\cdot)$ on $[-R, R]$ is given. For each non-negative integer n, let*

$$B_n(x) := L^n B\Big(\frac{x}{L^n}\Big).$$

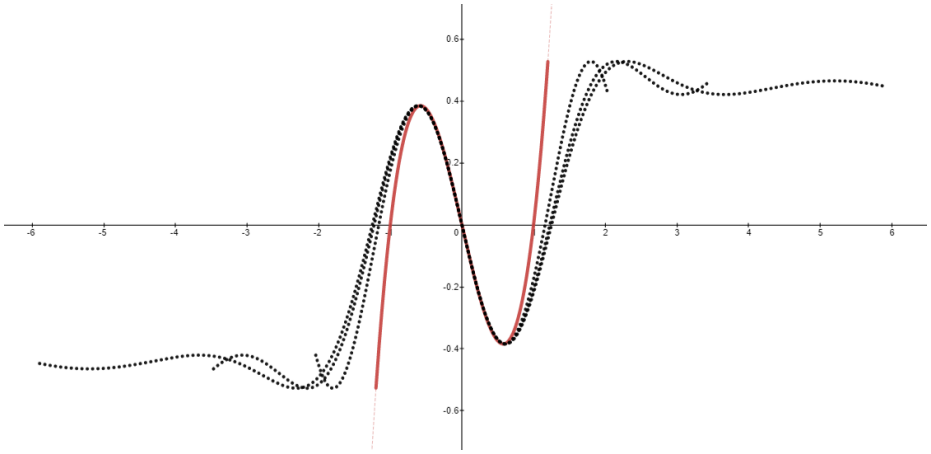


Fig. 4: The functions with extended valid intervals after $1, 2, 3$ iterations of domain-extension method to the red curve. Note that the feature of the original function is preserved on $[-0.5, 0.5]$

Then, after n sequential composition of $B_i(\cdot)$ to $p(\cdot)$,

$$\left\| p \circ B_0 \circ \cdots \circ B_{n-1} \right\|_{\infty,[-L^n R, L^n R]} \leq \|p\|_{\infty,[-R,R]},$$

and

$$\left\| p - p \circ B_0 \circ \cdots \circ B_{n-1} \right\|_{\infty,[-r,r]} \leq M r^3 \frac{L^2}{L^2-1} \delta$$

holds where $M = \sup\{|p'(x)| : -r \leq x \leq r\}$.

Proof. By Theorem 2, $B_n(\cdot) \in \mathscr{D}\left(\frac{\delta}{L^{2n}}, r, L^n R, L^{n+1} R\right)$. By using Theorem 1, we conclude that

$$\begin{aligned}
&\left\| p - p \circ B_0 \circ \cdots \circ B_{n-1} \right\| \\
&\leq \|p - p \circ B_0\| + \sum_{i=1}^{n-1} \|p \circ B_0 \cdots \circ B_{i-1} - p \circ B_0 \cdots \circ B_i\| \\
&\leq M r^3 \delta \sum_{i=0}^{n-1} \frac{1}{L^{2i}} \ \leq M r^3 \delta \sum_{i=0}^{\infty} \frac{1}{L^{2i}} = M r^3 \frac{L^2}{L^2-1} \delta
\end{aligned}$$

Here, we denoted $\|\cdot\|_{\infty,[-r,r]}$ as $\|\cdot\|$. □

Theorem 3 implies that if we use a DEP with small δ, the output polynomial of domain-extension methodology has only a small difference to the input polynomial on $[-r, r]$, and at the same time, the output polynomial is bounded on wide interval, $[-L^n R, L^n R]$.

To briefly sum up, we suggest DEPs and domain-extension methodology with it that extends the interval type II of a polynomial efficiently while preserve the feature on interval type I. Note that in order to extend a polynomial's interval type II by a factor of L, domain-extension methodology uses $O(\log L)$ operations.

Remark 1. Note that the domain-extension methodology is done by invoking a simple base DEP repeatedly. Once we are given an appropriate base DEP $B(\cdot)$ that can be stably evaluated based on HE (e.g., it does not considerably magnify the errors accompanied by HE), its scaled function $B_n(x)$ would be also HE friendly. This makes domain-extension methodology stable and HE friendly.

Example 1. $d(x) = -\frac{4}{27}x^3 + x$ is a DEP in $\mathscr{D}\left(\frac{4}{27}, r, 1, L\right)$ for each $1.5 < L < 1.5\sqrt{3}$ and $0 < r < 1.5$ such that $d(r) < d(L)$. Note that Theorem 2 implies that we can decrease δ as much as possible by scaling.

4 Privacy-preserving Computation Over Wide Input Range

In this section, we explain how domain-extension methodology can be used for privacy-preserving computations based on HE. We remark that in order to exploit HE, non-polynomial functions should be replaced by their polynomial approximations, and the approximation interval should be large enough to comprise all input values. Since polynomial approximation on wide interval in general accompanies high degree and considerable computational overheads, domain-extension methodology may provide a more efficient solution to manage a wide interval.

We suggest two applications of domain-extension methodology. First, we show that for several functions (e.g., the logistic, arctan, tanh, capped ReLU, and Gaussian function), domain-extension methodology extends not only interval type II, but also interval type I. As a consequence, by using domain-extension methodology, we can evaluate those functions based on HE on wide domain intervals. We stress out that this is substantially more efficient than other known polynomial approximation techniques.

Second, we explain that domain-extension methodology can effectively manage rare outlying data. Suppose there are some rare outliers while the most of data are from a narrow domain interval. The known polynomial approximations soar rapidly on the out of fixed approximation interval, so even a few outliers have a potential to ruin the ciphertext. More precisely, once the evaluation value does not belong to the plaintext space of HE, the ciphertext will be polluted, and the entire results will be contaminated. Hence, it is inevitable to use the polynomial approximation on significantly large interval, which is too costly to evaluate in encrypted state. We suggest an efficient solution to address this issue by using domain-extension methodology in Section 4.2.

4.1 Uniform Approximation using DEPs

We introduce an efficient polynomial approximation technique for functions that converges at infinities (i.e., functions in $C_{\lim}$). We first remark that there are a number of ML algorithms utilizes functions that converges at infinities; e.g., the logistic function, tanh function, arctan, capped ReLU function [21], and Gaussian function. To perform those ML algorithms in encrypted state, it is essential to uniformly approximate by polynomials the functions in $C_{\lim}$. However, with known approaches, the HE evaluation of such functions on sufficiently large intervals accompanies a considerable overhead.

By utilizing our domain-extension method in Section 3, we suggest an efficient polynomial approximation technique for functions in $C_{\lim}$. The idea is simple: find an approximation on a smaller interval and extend its approximation interval by using domain-extension method. More precisely, we approximately compute $f(\cdot) \in C_{\lim}$ on $[-R, R]$ with $O(\log R)$ multiplications. Algorithm 1 describes how to utilize domain-extension methodology for this scenario.

Input: $f(\cdot) \in C_{\lim}, x \in [-L^n r, L^n r]$.
Output: Approximate value of $f(x)$.
1: $P(\cdot) :=$ the minimax polynomial on $[-r, r]$
2: Take $B(\cdot) \leftarrow \mathscr{D}(\delta, r, R, LR)$
3: $y = \frac{x}{r}$
4: **for** $i \leftarrow n$ **to** 0 **do**
5: $\quad y = L^i B\left(\frac{y}{L^i}\right)$
6: **end for**
7: $y = P(ry)$
8: **return** y

Algorithm 1: Approximate evaluation of C_{lim} functions on wide interval

As in Figure 4, our domain-extension methodology stretches the graph of given polynomial by pulling on both ends while fixing the central part (interval type I) of the original function. Thus, in the case of approximate polynomials of $f(\cdot) \in C_{\lim}$, domain-extension method immediately extends not only interval type II but also interval type I, so we yield an efficient polynomial approximation $f(\cdot)$ on the extended wide interval. More precisely, Theorem 4 states that Algorithm 1 is correct unless r is too small.

Theorem 4. *Assume that a function $f(\cdot) \in C_{\lim}$ and a DEP $d(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ such that*

$$|f(x) - f(y)| < \epsilon_1 \quad \text{if } x, y \le -d(r)$$
$$|f(x) - f(y)| < \epsilon_1 \quad \text{if } x, y \ge d(r).$$

for some $\epsilon_1 > 0$. Suppose that $p(\cdot)$ on $[-R, R]$ is a polynomial approximation of $f(\cdot)$ that satisfies

$$|f(x) - p(x)| < \epsilon_2 \quad \textit{if } x \in [-R, R].$$

for some $\epsilon_2 > 0$.

Then, $p \circ d(\cdot)$ is a polynomial approximation of $f(\cdot)$ on $[-LR, LR]$ with maximum error is bounded by $max(Mr^3\delta, \epsilon_1) + \epsilon_2$.

Moreover, after the Algorithm 1, we get the polynomial approximation on $[-L^nR, L^nR]$ with maximum error less than
$max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) + \epsilon_2$.

Proof. From Theorem 1, for each $x \in [-r, r]$,

$$\begin{aligned} |p \circ d(x) - f(x)| &\leq |p \circ d(x) - p(x)| + |p(x) - f(x)| \\ &\leq Mr^3\delta + \epsilon_2 \leq max(Mr^3\delta, \epsilon_1) + \epsilon_2. \end{aligned}$$

Meanwhile, for each $x \in [r, LR]$,

$$\begin{aligned} |p \circ d(x) - f(x)| &\leq |p \circ d(x) - f \circ d(x)| + |f \circ d(x) - f(x)| \\ &\leq \epsilon_1 + \epsilon_2 \leq max(Mr^3\delta, \epsilon_1) + \epsilon_2. \end{aligned}$$

And the same argument holds for $x \in [-LR. - r]$. Thus,

$$|p \circ d(x) - f(x)| \leq max(Mr^3\delta, \epsilon_1) + \epsilon_2$$

for all $x \in [-LR, LR]$.

Finally, the similar argument with Theorem 3 proves that Algorithm 1 serves the uniform approximation on $[-L^nR, L^nR]$ with maximum error less than $\max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) + \epsilon_2$. □

As as a consequence, we can approximately evaluate a function $f(\cdot) \in C_{\lim}$ on arbitrarily large interval $[-R, R]$ within the error less than $\epsilon = \max(\frac{Mr^3L^2}{L^2-1}\delta, \epsilon_1) + \epsilon_2$ by using Algorithm 1 with $O(\log R)$ iterations of domain-extension processes.

Now we measure the costs of our algorithm for HE computation. We measure the computational cost in terms of the number of multiplications during the computation, because multiplication is much heavier than other operations in HE. To approximately evaluate a function in $f(\cdot) \in C_{\lim}$ on sufficiently large interval $[-R, R]$, we perform domain-extension process for $O(\log R)$ times; hence, our method uses $O(\log R)$ multiplications in encrypted state. Also, we point out that our method exploits $O(1)$ memory since the domain-extension processes are sequentially executed.

For the computation based on HE, the multiplicative depth is also an important factor since it determines the number of required bootstrapping during the computation. In the term of multiplicative depth, our method consumes $O(\log R)$ multiplicative depth since we invoke DEP for $O(\log R)$ times.

Remark 2. Our algorithm can be easily extended to a function whose difference from a low degree polynomial converges to constants at infinities. Note that such function can be represented as $f(x)+p(x)$, where $f(\cdot) \in C_{\lim}$ and $p(\cdot)$ is a lower degree polynomial.

Comparison to the previous approaches For the privacy-preserving evaluation of non-polynomial functions on wide domain intervals, in the best of our knowledge, there was no known polynomial approximation technique optimized for HE; instead, all previous works used general polynomial approximation techniques. For given maximum error, minimax approximation technique serves the polynomial approximation with the smallest degree, so it has been prevalent to use minimax approximation for HE. To evaluate the minimax approximation, Paterson-Stockmeyer algorithm is being widely exploited since it uses a less number of multiplications. In the best of our knowledge, minimax approximation with Paterson-Stockmeyer algorithm is currently most efficient way to evaluate a function on wide interval based on HE.

However, as we illustrate in Figure 5 and prove in Theorem 5, to approximately compute $f(\cdot) \in C_{\lim}$ on $[-R, R]$ in encrypted state with minimax approximation, we should compute polynomial of degree $\Omega(R)$. As a consequence, even with Paterson-Stockmeyer algorithm, $\Omega(\sqrt{R})$ multiplications, and it consumes $\Omega(\sqrt{R})$ memory space.

Theorem 5. *Assume that we are given a function $f(\cdot) \in C_{\lim}$ where $\lim_{x\to\infty} f(x) \neq \lim_{x\to-\infty} f(x)$. For a sufficiently small $\epsilon > 0$, the minimax polynomial approximation on $[-R, R]$ with maximum error ϵ has degree $\Omega(R)$ as $R \to \infty$. Empirically, the degree is $O(R)$ as $R \to \infty$ even if $f(x) \neq \lim_{x\to-\infty} f(x)$ unless $f \equiv 0$.*

Proof. Without loss of generality, assume that $\lim_{x\to-\infty} f(x) = 0$ and $\lim_{x\to\infty} f(x) = 1$. There exists $r > 0$ such that $|f(x) - 1| < \epsilon/2$ if $x > r$, and $|f(x)| < \epsilon/2$ if $x < -r$ For sufficiently large $R \gg r$, let $P_R(\cdot)$ be the minimax polynomial approximation of $f(\cdot)$ on R by maximum error less than $\epsilon/2$. Let d be the degree of P_R. Then,

$$\begin{aligned}
\epsilon &\geq \sup\{|P_R(x) - sgn(x)| : x \in [-R, -r] \cup [r, R]\} \\
&= \sup\{|P_R(rx) - sgn(x)| : x \in [-\frac{R}{r}, -1] \cup [1, \frac{R}{r}]\} \\
&\geq \sqrt{\frac{2}{\pi A d}}(A-1)\Big(\frac{A-1}{A+1}\Big)^{\frac{d-1}{2}} \sim \sqrt{\frac{2}{\pi}}\Big(\sqrt{\frac{R}{dr}} - \sqrt{\frac{dr}{R}}\Big)
\end{aligned}$$

where $A = R/r$. This comes from the arguments in [22].
This implies that $\sqrt{\frac{R}{dr}} < E$ for some constant $E(\epsilon)$. As a consequent, $d > CR$ for some constant $C(\epsilon)$, and $d = \Omega(R)$. The graphs in Figure 5 shows that the empirical result. □

Compared to the minimax approximation, our approach is more efficient in terms of both computational and memory costs. To approximately evaluate on $[-R, R]$ based on HE, our method uses $O(\log R)$ multiplications while minimax approximation uses $\Omega(\sqrt{R})$ multiplications. In the term of multiplicative depth, both our method and the minimax approximation consumes $O(\log R)$ multiplicative depth for the evaluation. Thus, our method is computationally more efficient than minimax approximation to be evaluated based on HE. See Figure 6 for the experimental results.

Also, we point out that our method uses $O(1)$ memory during the computation while the minimax approximation uses $\Omega(\sqrt{R})$ memory for the Paterson-Stockmeyer algorithm. Table 1 summarizes the costs to approximate a function on $[-R, R]$ for a fixed uniform error by using each of minimax approximation and our algorithm.

One another point is that our method provides a stable evaluation even for large domain intervals. This is due to the fact that during the domain-extension, all we compute is DEPs of low degrees. On the other hand, when we exploit the minimax approximation with Paterson-Stockmeyer algorithm, we should precisely evaluate the Chebyshev polynomial of degree $\sqrt{d}$ where $d = \Omega(R)$. Consequeuntly, as R grows, the error accompanied by HE would make it challenging to use Paterson-Stockmeyer algorithm.

Also, our algorithm is much simpler to implement. In contrast to our algorithm, finding the minimax polynomial on significantly wide interval is nontrivial in general.

	# of mult	mult. depth	memory
Minimax	$\Omega(\sqrt{R})$	$\Omega(\log R)$	$\Omega(\sqrt{R})$
Ours	$O(\log R)$	$O(\log R)$	$O(1)$

Table 1: Cost of our algorithm and minimax approximation to approximate on $[-R, R]$ for a fixed uniform error.

Remark 3. Once regard Algorithm 1 as a polynomial, the degree of the polynomial is $O(R^c)$ for some constant c. The computational efficiency of our algorithm comes from the fact that our algorithm can be computed with a less number of multiplications while there is no known method to efficiently compute the minimax polynomial of degree $\Omega(R)$.

Further Optimizations We can further reduce the error accompanied by the domain-extension methodology. Suppose we want to approximate a function $f(\cdot) \in C_{\lim}$ on a large domain interval $[-R, R]$; by using DEP $d(\cdot)$ and a polynomial approximation on the small interval $[-r, r]$.

Instead of using the minimax approximation $P(\cdot)$ of $f(\cdot)$ on $[-r, r]$, we can reduce the error by using the minimax approximation $Q(\cdot)$ of $f \circ d^{-1}(\cdot)$ on $[-r, r]$ with the same degree of $P(\cdot)$; i.e., $Q(x) \approx f \circ d^{-1}(x)$ where $d^{-1} \circ d(x) \approx x$. Note that

$$\begin{aligned} \|f - P \circ d\|_{\infty,[-R,R]} &\geq \|f \circ d^{-1} - P \circ d \circ d^{-1}\|_{\infty,[-r,r]} \\ &\approx \|f \circ d^{-1} - P\|_{\infty,[-r,r]} \\ &\geq \|f \circ d^{-1} - Q\|_{\infty,[-r,r]} \\ &\approx \|f - Q \circ d\|_{\infty,[R,R]}. \end{aligned}$$

To sum up, we can expect to yield a better uniform error than that in Theorem 5 by considering the minimax approximation of $f \circ d^{-1}(\cdot)$.

4.2 Accommodation of Outliers

In this section, we utilize DEPs to accommodate rare outliers distributed on a wide interval. A polynomial approximation has weakness to the outliers from the outside of the approximation interval, because its value soars rapidly on the inputs from the out of the approximation interval. This can be a serious issue since it has a potential to damage the ciphertext and ruin the entire computation. To take the training phase of neural networks in encrypted state as an example, a single strange training datum can generate an input of an activation function from out of the approximation interval, and its polynomial evaluation might be out of the plaintext space of HE. As a result, a single outlying datum is able to destroy the ciphertext, and ruin the entire training phase. Moreover, all the process are being done in encrypted state, so we even cannot detect which datum caused the failure.

To address this issue, there should be an clever way to manage such outlying inputs from wide intervals. However, it is impractical to use a polynomial approximation on huge approximation intervals to manage all such outliers. We suggest, instead, to consider a polynomial approximation that is accurate on a relatively small interval (interval type I) and *bounded on a huge interval* (interval type II) at the same time. The rare outliers from the huge interval may not produce meaningful results; albeit, they do not damage neither the ciphertext nor other parts of the algorithm. For example, in the case of neural network training on encrypted data, we now can prevent abnormal data from contaminating the entire process.

For the formal description, we consider $\mathscr{F}$, a class of approximate functions of $f(\cdot)$, as followings. In high level, this is a class of approximate polynomials that is accurate on small interval $[-r, r]$ by error ϵ, and bounded by ρ on the large interval $[-R, R]$.

Definition 2. *For a given function $f(\cdot)$ on $[-R, R]$ and given $R > \rho > r > 0$ with small $\epsilon > 0$, we define $\mathscr{F}(f; \epsilon, r, \rho, R)$ to be a class of function of $p(x)$ satisfying:*
(a) $|p(x) - f(x)| < \epsilon \ \forall x \in [-r, r]$

(b) $|p(x)| < \rho \ \forall x \in [-R, R]$.

$\mathscr{F}(f; \epsilon, r, \rho, R)$ is a class of functions that is accurately approximates $f(\cdot)$ on $[-r, r]$ and is bounded by ρ on $[-R, R]$. Thus, the HE evaluation on $[-r, r]$ would be valuable, and that on $[-R, R]$ would be stable (i.e., each function value on $[-R, R]$ belongs to the plaintext space of HE).

In this case, our domain-extension methodology can be applied; as we regard $[-r, r]$ as the interval type I and $[-R, R]$ as the interval type II. Theorem 3 explains how domain-extension methodology enables us to efficiently increase the stable interval. Note that Theorem 6 is an immediate outcome of Theorem 3.

Theorem 6. *Assume that we are given a function $f(\cdot)$ and its approximation $p(\cdot) \in \mathscr{F}(f; \epsilon, r, \rho, R)$; also, suppose we are given a DEF $d(\cdot) \in \mathscr{D}(\delta, r, R, LR)$ where $LR > R > \rho > r > 0$ and $\delta, \epsilon > 0$ are small. Then,*

$$p \circ d(\cdot) \in \mathscr{F}\Big(f; \epsilon + Mr^3\delta, r, \rho, LR\Big)$$

where $M = sup\{|p'(x)| : -r < x < r\}$. Moreover, if we let

$$B_n(x) := L^n d\Big(\frac{x}{L^n}\Big)$$

for each non-negative integer n, then

$$F \circ B_0 \circ \cdots \circ B_{n-1}(\cdot) \in \mathscr{F}\Big(f; \epsilon + Mr^3\frac{L^2}{L^2-1}\delta, r, \rho, L^n R\Big).$$

To put it all together, when we use a polynomial approximation in $\mathscr{F}$, we can manage the outliers from a wide interval $[-R, R]$ by using additional $O(\log R)$ number of operations.

5 Privacy-preserving Logistic Regression

In this section, we implement our domain-extension methodology, and apply it for the privacy-preserving logistic regression based on HE. Note that training and inference of a logistic regression model can be done by the combination of linear operations and the logistic function.

The logistic function is in C_{lim}, so domain-extension methodology provides an efficient uniform approximation of the logistic function on wide intervals (see Section 4.1). We implement our domain-extension methodology for the logistic function, and by using it, we perform the logistic regression based on HE. We adopt CKKS scheme [12, 13] among many HE schemes, since it supports floating point operation on real number.

We introduce three experiments: (1) the comparison of our method to the minimax approximation for the evaluation of the logistic function based on HE, (2) the privacy-preserving logistic regression on MNIST with our method, and (3) the privacy-preserving logistic regression on Swarm Behavior dataset with our method.

In the first experiment, we implement and compare our domain-extension methodology on the logistic function to the current best approach: minimax approximation and Paterson-Stockmeyer algorithm. In the second and third experiments, by using our method, we perform logistic regression training on two public datasets: MNIST and Swarm Behavior dataset.

In the case of MNIST dataset, all data belong to a bounded interval, $[0, 255]^{28\times 28}$ or $[0, 1]^{28\times 28}$, so we select the sufficiently wide input domain of the logistic function so that is capable to contain all the possible outliers. It is true that the most of inputs of logistic functions come from narrower domain intervals, but the calculated wide approximation interval theoretically guarantees the success of the HE computation.

On the other hand, the values of Swarm Behavior data are not explicitly bounded, so we cannot calculate the domain interval of the logistic function that can contain all the possible outliers. Instead, we train the model by using the approximation of the logistic function on sufficiently large input interval. Moreover, we monitored the input values of logistic functions during the training in unecnrypted state, and we observed that the size of input values grows bigger than 1333. This shows that it is necessary to approximate the logistic function on wide intervals to train logistic regression model on many of public datasets including Swarm Behavior dataset. In this work, we utilize a polynomial approximation on sufficiently large approximation interval, $[-7683, 7683]$, for Swarm Behavior dataset.

In this paper, we use relative error to measure the difference between two logistic regression models. More precisely, for to models w_p and w_c (e.g., each of model trained on plaintext and ciphertext), we measure the difference between them by $\frac{\|w_p - w_c\|}{\|w_p\|}$ in L^2-norm.

5.1 Approximation of the Logistic Function

In this experiment, we demonstrate that our method is substantially more efficient compared to the minimax approach to evaluate the logisic function on large domain interval.

For the comparison, we evaluate the logistic function based on HE on wide interval by using two different methods: (1) our method and (2) minimax approximation with Paterson-Stockmeyer algorithm. Under fixed parameters including the maximum norm error, we measure the runtime of both methods for the evaluation on various sizes of domain intervals.

Parameter Selection In this experiment, for both methods, we measure the runtime for the HE evaluation of polynomial approximations with maximum norm error less than 0.045.

- For the minimax approximation, we use the minimax polynomial with the smallest degree among the minimax polynomials with maximum norm error

less than 0.045, and we evaluate the minimax polynomial with Paterson-Stockmeyer algorithm with Chebyshev basis [8]. To find the minimax polynomials, we used chebfun library. [17]

- For our domain-extension method, we used a DEP $B(x) = x - \frac{4}{27}x^3$ with extension factor 2.45. We extend the domain interval of $P(x)$, the degree 9 minimax polynomial of the logistic function on $[-14.5, 14.5]$. Note that the maximum errors of extended polynomial approximations are all less than 0.045.
- We measured the evaluation time of each method under various sizes of the domain intervals, from $[-14.5, 14.5]$ to $[-3136, 3136]$, increasing 2.45 times each.
- For CKKS parameters, we took $N = 2^{16}$ with initial ciphertext modulus $q_L = 2^{1081}$. Hamming weight of the secret polynomial is set to 128. Note that these parameters achieve 128-bits security [23–25].

Environment All experiments were performed on Intel Xeon CPU E5-2620 v4 at 2.10GHz processor. We used a single thread for the experiments.

Result The experimental result is represented in Figure 6. It shows the runtime in seconds of each method to evaluate the logistic function on interval $[-x, x]$. Note that The x-axis is logarithmic scaled.

We first highlight that our method is more efficient than the minimax approach. To uniformly approximate the logistic function on $[-87, 87]$, our method is more than 2.34× faster than the minimax approach (ours consumed 6.34 seconds while the minimax approach used 14.89 seconds). In the case of approximation on $[-1279, 1279]$, ours consumed 9.14 seconds, and it is 8.7× faster than the minimax approach which consumed 79.56 seconds.

Moreover, our method tends to asymptotically faster than the minimax approach. As in Figure 6, we can observe that the runtime for the minimax approach increases subexponentially with respect to $\log R$, where R is the size of the domain interval. On the other hand, the runtime for our method increases linearly with respect to $\log R$. This matches to our expectation in Section 4.1.

Meanwhile, our method use a bit larger multiplicative depth than the minimax approach; for example, in the case of experiment on $[-3136, 3136]$, the multiplicative depth of our method is 14 while that for minimax approach is 10. However, the multiplicative depth for both methods are asymptotically same (see Section 4.1).

Our method provides an easy implementation with reasonable precision bits. In contrast, the minimax approximation uses the coefficients with high precision, and in our implementation, it fails to work for the domain interval larger than $[-1279, 1279]$. In Figure 6, we extrapolate the runtime larger than $[-1279, 1279]$ assuming the cost is $O(\sqrt{R})$ (see Section 4.1).

5.2 Logistic Regresssion on MNIST Dataset

Dataset description MNIST is a dataset of handwritten digits, which contains 60000 training samples and 10000 test samples [26]. In this experiment, we selected the samples with two labels, 3 and 8, to perform the binary classification using logistic regression. For the training set, the first 9600 training samples with label of either 3 or 8 were selected. For the test set, all test samples with label of either 3 or 8 were selected.

Selection of Approximation Interval All samples of MNIST belong to $[0,1]^{28\times 28}$. Consequently, we can calculate the upper bound of the size of the norm of the trained weight by logistic regression for each iteration. The polynomial approximations on such domain intervals guarantee the success of privacy-preserving logistic regression training without exceeding the domain of the polynomial approximation.

Let $w_t \in \mathbb{R}^D$ be trained weight vector(including bias component) by logistic regression, after t iterations. Also α be the learning rate, n be the mini-batch size, $D = d+1$, d be the number of attributes of the sample, and $x \in \mathbb{R}^D$ be an arbitrary single sample which satisfies that every component belongs to $[0,1]$. Note that the last component of x has always value 1, since we assume that bias is in the last component of the weight vector. $y \in \{-1,1\}$ is the corresponding label of x.

We will find M such that $|w_t \cdot x| \leq M$ for arbitrary x. By logistic regression training, weight vector is updated as follows [2, 20]:

$$w_{t+1} = w_t + \frac{\alpha}{n}\sum_{i=1}^{n} y\sigma(-y_i(w_t \cdot x_i))x_i.$$

Thus, the following holds.

$$\begin{aligned}
\|w_{t+1}\|^2 &= \left\| w_t + \frac{\alpha}{n}\sum_{i=1}^{n} y_i\sigma(-y(w_t \cdot x_i))x_i \right\|^2 \\
&= \|w_t\|^2 + \frac{2\alpha}{n}\sum_{i=1}^{n}\sigma(-y_i(w_t \cdot x_i))(y_i(w_t \cdot x_i)) \\
&+ \frac{\alpha^2}{n^2}\sum_{i,j} y_i y_j \sigma(-y_i(w_t \cdot x_i))\sigma(-y_j(w_t \cdot x_j))(x_i \cdot x_j) \\
&\leq \|w_t\|^2 + 0.6\alpha + \alpha^2 D \ (\because x\sigma(-x) \leq 0.3 \ \forall x \in \mathbb{R})
\end{aligned}$$

Note that every norm below are Euclidean norm. Finally, mathematical induction shows that

$$\|w_t\|^2 \leq t(\alpha^2 D + 0.6\alpha),$$

and

$$|w_t \cdot x|^2 \leq \|w_t\|^2\|x\|^2 \leq t(\alpha^2 D + 0.6\alpha)D < t(\alpha D + 0.3)^2$$

# of iter	accuracy (0.1)	maxinput (0.1)	accuracy (1.0)	maxinput (1.0)
3	91.12%	1.22	50.90%	38.3
6	89.16%	2.06	85.13%	25.2
9	92.13%	2.37	93.85%	16.3
12	92.13%	3.00	94.35%	17.0
15	92.33%	3.35	94.55%	16.1
18	93.29%	3.60	94.70%	16.3
21	93.80%	3.39	96.06%	12.1
24	94.15%	4.02	96.11%	14.8
27	94.15%	3.69	95.61%	12.5
30	94.65%	4.17	96.11%	13.4

Table 2: The result of MNIST dataset training with learning rate 0.1 and 1.0.

hold for each t. Now we get $|w_t \cdot x| < \sqrt{t}(\alpha D + 0.3)$. Therefore, the input of the logistic function in t th iteration must belong to

$$[-\sqrt{t}(\alpha D + 0.3), \sqrt{t}(\alpha D + 0.3)]. \tag{1}$$

Parameter selection We used the parameters for CKKS scheme, logistic regression, and our domain-extension method as follows.

- For CKKS parameters, we took $N = 2^{17}$ with initial ciphertext modulus $q_L = 2^{2150}$. Hamming weight of the secret polynomial is set to 128. Note that these parameters achieve 128-bits security [23–25].
- We used the mini-batch stochastic gradient descent with mini-batch size 320. We iterated 30 times, which is 1 epoch. We used two learning rates, 0.1 and 1.0, and compared two results.
- For the polynomial approximation of the logistic function, we used our domain-extension method. We use the DEP $B(x) = x - \frac{4}{27}x^3$ with extension factor 2.45. We extend the domain interval of $P(x)$, the degree 15 minimax polynomial of the logistic function on $[-14.5, 14.5]$.
- We approximate the logistic function on sufficiently large domain by using Equation 5.2. For example, in the case of learning rate 1.0, we approximated the logistic function maximally on $[-7683, 7683]$, which can be done by 7 domain-extension processes.

Environment All experiments were performed on Intel Xeon CPU E5-2620 v4 at 2.10GHz processor. Also, 8 threads were used for the experiments.

Results Table 2 shows the experimental results. Each row describes the state of logistic regression model after given number of iteration. The second and forth columns, (accuracy(0.1), accuracy(1.0)), indicate the accuracy of the (encrypted) logistic regression model with learning rate 0.1 and 1.0 respectively. The third and fifth columns, (maxinput (0.1), maxinput (1.0)), show the maximum absolute value of inputs of the logistic function during the iteration of training. The result shows us two points as follows.

- Appropriate large learning rate makes training faster. However, large learning rate also increases the input values of the logistic functions during training, so too narrow approximation interval (e.g., $[-8, 8]$ as in the previous works or less) would restrict us to select the optimal hyper-parameter.
- We cannot ensure that the input value for the logistic function varies gently. As seen in our experiment with learning rate 1.0, maxinput value varies largely at the early stage of the training, even though maxinput value became about $10-15$ finally. This shows us that just moderately extending the interval of approximation (such as $[-8, 8]$ to $[-16, 16]$) cannot be a fundamental solution.

5.3 Swarm Behavior Dataset

Dataset description Swarm Behavior Dataset is a dataset of determining 3 behaviors of each swarm sample. The dataset contains 24016 swarm samples, which has 3 binary behavior labels that indicate whether the swarm is (1) aligned, (2) flocking, and (3) grouped. Each swarm comprises 200 individuals, and each individual is characterized by 12 features such as position and velocity; in other words, each sample swarm has 2400 real-valued features.

In this experiment, we solve the binary classification of 'aligned' label by using logistic regression. For the training set, we randomly pick 3840 samples among 24016 samples, and the other samples are used for the test set.

Parameter selection The parameters for CKKS scheme, logistic regression, and approximation of the logistic function are given here. The number of extension is also given, since the value of data are not bounded and we should select the number of extension.

- For CKKS parameters, we took $N = 2^{17}$ with initial ciphertext modulus $q_L = 2^{2150}$. Hamming weight of the secret polynomial is set to 128. Note that these parameters achieve 128-bits security [23–25].
- We used the mini-batch stochastic gradient descent with mini-batch size 240. We iterated 16 times, which is 1 epoch. We set 10^{-6} for learning rate.
- For the polynomial approximation of the logistic function, we used our domain-extension method. We use the DEP $B(x) = x - \frac{4}{27}x^3$ with extension factor 2.45. We extend the domain interval of $P(x)$, the degree 15 minimax polynomial of the logistic function on $[-14.5, 14.5]$. We utilize 7 times of domain-extension processes, so the approximation interval is about $[-7683, 7683]$.

# of iterations	accuracy (enc)	accuracy (plain)	relative error	max input
1	78.60%	78.60%	0.05%	0.0
2	82.90%	83.05%	1.32%	10.0
3	85.88%	85.88%	1.83%	72.0
4	88.52%	88.53%	1.81%	6.3
5	89.60%	89.72%	4.47%	86.2
6	90.38%	90.39%	3.50%	40.6
7	92.62%	92.65%	5.65%	1358.5
8	93.11%	93.15%	4.69%	154.3
9	93.27%	93.27%	6.54%	353.2
10	94.01%	94.02%	6.37%	33.4
11	94.67%	94.61%	6.25%	129.6
12	94.52%	94.51%	6.13%	284.6
13	94.69%	94.71%	5.67%	176.8
14	95.15%	95.14%	8.32%	1221.2
15	95.55%	95.51%	7.89%	1843.3
16	95.78%	95.72%	7.52%	26.6

Table 3: The result of Swarm Behavior dataset training.

Environment All experiments were performed on Intel Xeon CPU E5-2620 v4 at 2.10GHz processor. Also, 8 threads were used for all experiments here.

Results Table 3 shows the experimental results of logistic regression training on Swarm Behavior dataset. Each row describes the trained model after given number of iteration. The second and third columns show the accuracy of the model trained in encrypted and unencrypted state respectively. The forth column, relative error, indicates the difference between two models that are trained in encrypted and unencrypted states. Finally, the last column, max input, is the maximum size of inputs of the logistic function during the training iterations in encrypted state. We point out two points as follows.

- Our approximation of the logistic function on wide interval behaves similar to the exact logistic function. In particular, the accuracy of HE-based trained model is almost same to the plaintext-based trained model. In particular, we can observe that the relative error between the models is small.
- The polynomial approximation on substantially wide approximation interval is necessary. There are several large inputs of the logistic function during the training, so the polynomial approximation of the logistic function on a wide interval (e.g., $[-7000, 7000]$) has a key role.

6 Conclusion

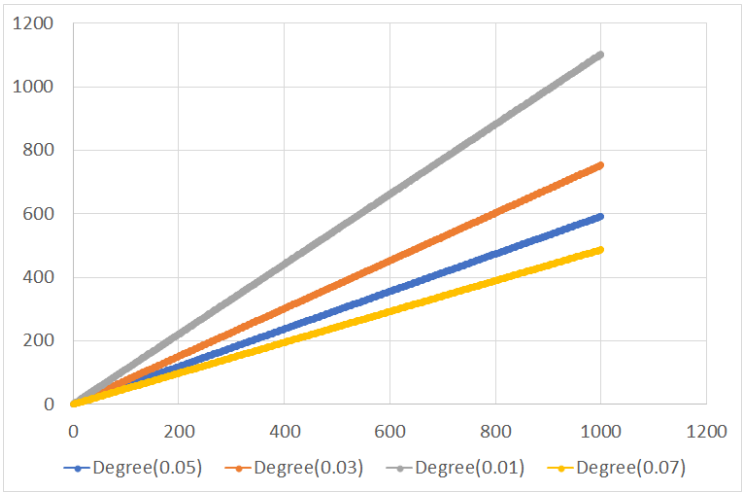
In this work, we propose a new efficient method for HE evaluation on large intervals. For the stable computation based on HE, an efficient method for HE evaluation on large interval is crucial. For instance, to train logistic regression model based on HE, it is necessary to approximate the logistic function on a large interval; e.g, larger than $[-1843, 1843]$ for the Swarm Behavior dataset.

Our method evaluates on a large domain interval $[-R, R]$ a function that converges at infinities, with $O(\log R)$ multiplications and $O(1)$ space. This is both asymptotically and practically more efficient than the previous best approach, the minimax approximation and Paterson-Stockmeyer algorithm, which uses $O(\sqrt{R})$ multiplications and $O(\sqrt{R})$ space. Also, our algorithm can be easily extended to functions whose differences from a low degree polynomial converges to constants at infinities. We leave an efficient method for the HE evaluation of arbitrary functions on large interval as an open problem.

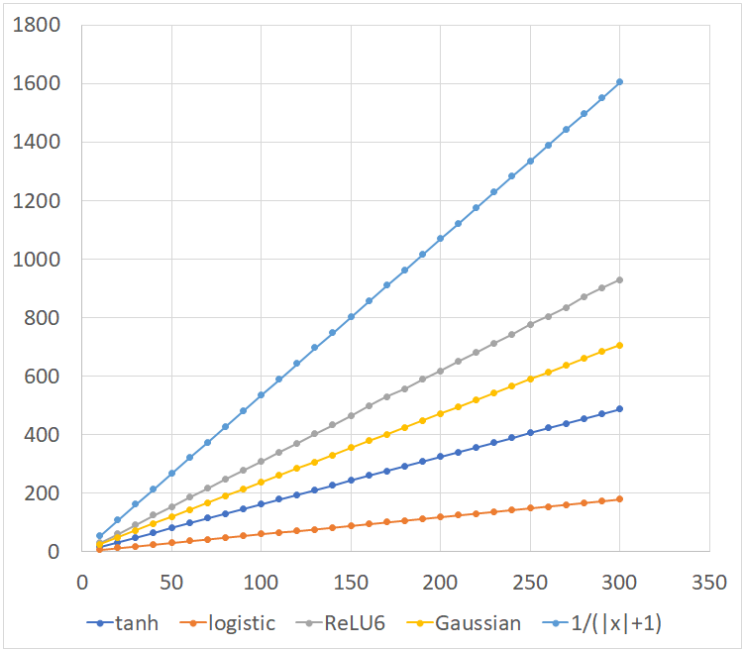
References

1. J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46 938–46 948, 2018.
2. A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon, "Logistic regression model training based on the approximate homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, p. 83, 2018.
3. H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter, "Logistic regression over encrypted data from fully homomorphic encryption," *BMC medical genomics*, vol. 11, no. 4, p. 81, 2018.
4. K. Han, S. Hong, J. H. Cheon, and D. Park, "Logistic regression on homomorphic encrypted data at scale," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 9466–9471.
5. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, 2016, pp. 201–210.
6. E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.
7. M. S. Paterson and L. J. Stockmeyer, "On the number of nonscalar multiplications necessary to evaluate polynomials," *SIAM Journal on Computing*, vol. 2, no. 1, pp. 60–66, 1973.
8. H. Chen, I. Chillotti, and Y. Song, "Improved bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 34–54.
9. J.-W. Lee, E. Lee, Y. Lee, Y.-S. Kim, and J.-S. No, "High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approxiamtion and inverse sine function," *IACR Cryptol. ePrint Arch*, vol. 2020, 2020.
10. J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," *IACR Cryptol. ePrint Arch*, vol. 2020, p. 1203, 2020.

11. J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1234, 2019.
12. J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.
13. J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 360–384.
14. J. H. Cheon, K. Han, S.-M. Hong, H. J. Kim, J. Kim, S. Kim, H. Seo, H. Shim, and Y. Song, "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE access*, vol. 6, pp. 24 325–24 339, 2018.
15. J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *International Conference on Selected Areas in Cryptography*. Springer, 2019, pp. 227–249.
16. R. Pachón and L. N. Trefethen, "Barycentric-remez algorithms for best polynomial approximation in the chebfun system," *BIT Numerical Mathematics*, vol. 49, no. 4, p. 721, 2009.
17. T. A. Driscoll, N. Hale, and L. N. Trefethen. (2014) Chebfun guide. Oxford.
18. C. S. Burrus, J. W. Fox, G. A. Sitton, and S. Treitel, "Horner's method for evaluating and deflating polynomials," *DSP Software Notes, Rice University, Nov*, vol. 26, 2003.
19. M. Fasi, "Optimality of the paterson–stockmeyer method for evaluating matrix polynomials and rational matrix functions," *Linear Algebra and its Applications*, vol. 574, pp. 182–200, 2019.
20. M. Kim, Y. Song, S. Wang, Y. Xia, X. Jiang *et al.*, "Secure logistic regression based on homomorphic encryption: Design and evaluation," *JMIR medical informatics*, vol. 6, no. 2, p. e8805, 2018.
21. A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
22. A. Eremenko and P. Yuditskii, "Polynomials of the best uniform approximation to sgn (x) on two intervals," *Journal d'Analyse Mathématique*, vol. 114, no. 1, p. 285, 2011.
23. J. H. Cheon, Y. Son, and D. Yhee, "Practical fhe parameters against lattice attacks," Cryptology ePrint Archive, Report 2021/039, 2021, https://eprint.iacr.org/2021/039.
24. M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.
25. M. R. Albrecht. (2017) A Sage Module for estimating the concrete security of Learning with Errors instances. https://bitbucket.org/malb/lwe-estimator.
26. Y. LeCun and C. Cortes, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/



(a) The minimal degrees for the minimax approximation to approximate the logistic function by fixed maximum norm errors on various domain sizes. The degree of the minimax polynomial increases linearly with respect to the size of domain.



(b) The minimal degrees for the minimax approximation to approximate the several functions by maximum errors of 0.05 on various domains. The degree of minimax polynomial increases linearly with respect to the size of approximation domain.

Fig. 5: The degree of minimax polynomial of functions in C_{lim} on various sizes of domains.

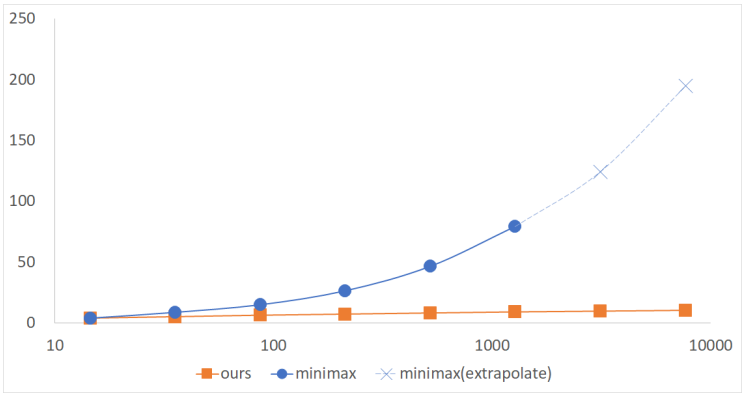


Fig. 6: Runtime(s) for the evaluation of the logistic function on various range of domain intervals using two different methods: minimax approximation and ours. The x-axis is logarithmic scaled.