Efficient Homomorphic Evaluation on Large Intervals

IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 17, 2022

< Introduction >

❖ Efficient Homomorphic(HE) Evaluation on Large Intervals

- **Problem Situation** [3p.]**:** 덧셈과 곱셈 연산만을 지원하는 특성에 기반한 동형암호 기반의 함수 평가 문제.

  → **비다항 함수**(예: 로지스틱) 의 **다항식 근사** 시 넓은 도메인에서 계산 비용 급증.

  → 연산 시의 값들이 HE의 plaintext 공간을 벗어나는 이상치 존재 시 암호문 연산이 제대로 이루어지지 않기에 큰 범위 설정 필요.

- **Existing Approaches** [12p.]**:** Minimax Polynomial 근사와 Paterson-Stockmeyer 알고리즘 사용.
  - 큰 도메인 설정 시 넓은 구간에서의 근사 정확도를 위해 높은 차수의 다항식 필요 ⇒ **계산 비용**이 급격히 높아짐.

- **Contribution:** ① **Domain-extension methodology :** Domain Extension Functions (DEFs), Domain Extension Polynomials (DEPs) 를 반복적으로 사용하여, 작은 구간의 다항식 근사를 넓은 구간으로 효율적으로 확장해 곱셈 횟수와 메모리 사용 최소화.

  ② **Uniform approximation on wide intervals :** 무한대에서 수렴하는 함수를 암호화된 상태에서 균일하게 근사.

  ③ **Accommodation of outliers from wide intervals :** 넓은 구간에서의 이상치 값을 포함할 수 있어, 동형 암호문 연산 결과의 안정성 유지 가능.

  ④ **Logistic regression over large datasets in encrypted state :** 충분히 큰 도메인에서(ex. $[-7683, 7683]$) 로지스틱 함수를 DEP로 근사하여 암호화된 상태에서 대규모 데이터셋에 대한 로지스틱 회귀 분류기 훈련 가능.

- **Result:** 큰 도메인 구간 $[-R, R]$ 에서 기존 방법 대비 계산 비용을 $\boldsymbol{O(\sqrt{R})}$**에서** $\boldsymbol{O(log\, R)}$**로 감소**시키면서 높은 정확도 유지 가능.

Introduction

❖ Homomorphic Encryption (HE)

- **덧셈**과 **곱셈**이라는 제한된 종류의 연산만을 지원하며, **평문 간의 연산을 암호화 한 상태**에서 할 수 있고
  이를 복호화했을 때 평문 간의 연산과 **동일한 결과**를 얻을 수 있는 기법
  → 데이터를 복호화하지 않고도 연산할 수 있어 **머신러닝에서의 프라이버시 보호**를 위한 기술로 이용할 수 있음.

- **Point 1)** Logistic 함수와 같은 non-polynomial 함수 연산 불가 ⇒ **범위를 정해 근사 다항식으로의 대체 필요**
- **Point 2)** 동형 암호 기반 연산 중에 값을 관찰할 수 없음 ⇒ **대체할 근사 다항식의 범위를 정하기 어려움**
- **Point 3)** 연산 시의 값들이 HE의 plaintext 공간을 벗어나면 암호문 연산이 제대로 이루어지지 않음 ⇒ **큰 범위 설정 필요**

“ HE를 이용한 넓은 범위에서의 비다항식 함수 근사 연산 방법 ”

Introduction

❖ Multiplication in HE

- HE 연산에서는 덧셈보다 **곱셈이 훨씬 비용이 크고**, 특히 두 개의 암호문 간 곱셈(Mult)은 암호문*평문 간의 곱셈보다 연산 비용이 크다.

  → 암호문 간 곱셈 시 두 다항식의 차수가 합산되어 암호문의 차원이 증가하고, 이때 고차원 암호문을 원래 차원으로 되돌리기 위해 Relinearization을 통해 암호문의 크기를 복원해 연산이 가능하도록 하는 등 추가적인 연산과 메모리 소모를 유발하기에 비용이 커짐.

- 곱셈이 반복될수록 **노이즈**가 증가하므로, 이는 정확한 복호화 결과 도출을 방해하기 때문에 일정 수준 이상이 되면 **Bootstrapping** 과정을 수행해야 한다.

  → Bootstrapping: 연산 중 증가하는 노이즈를 제거하여 레벨이 소진된 암호문을 복구함으로써 추가적인 연산이 가능하도록 만듦.

⇒ 따라서 HE 환경에서는 연산의 정확도를 유지하면서도 **곱셈 횟수를 줄이는 것이 최적화의 핵심**.

Related Works

❖ Related Works in Homomorphic Polynomial Evaluation

1) Machine Learning Over Encrypted Data

- Kim et al[2]: Least Square Fitting(최소 제곱 근사) 다항식을 이용하여 [-8,8] 구간에서 로지스틱 함수 근사.
- Chen et al[3]: Minimax 다항식 근사를 적용해 [-5,5]에서 로지스틱 회귀 수행. (입력 범위를 맞추기 위해 데이터 전처리 필요)

⇒ 근사 구간을 경험적으로 선택하였고, 넓은 구간에서 처리 시 cost가 매우 커지는 문제로 인해 대규모 데이터셋에 적용하기 어려움. 또한 근사 범위 내 입력값을 적용하기 위해 전처리하는 과정에서 정보 손실 위험 발생.

2) Computation With Less Multiplications

- 고차 다항식을 효율적으로 계산하는 Paterson-Stockmeyer 알고리즘 방식이 HE 기반 고차 다항식 평가에서 널리 사용됨.

⇒ 차수가 d인 다항식을 평가할 때 $\boldsymbol{O}(\sqrt{\boldsymbol{d}})$번의 곱셈만이 필요하지만, 넓은 구간에서 평가 시 여전히 많은 곱셈 필요.

3) Homomorphic Evaluation With Bit-Wise HE

- Bit-Wise HE: 각 비트를 개별적으로 암호화하여 덧셈, 곱셈의 산술 연산 시 계산 및 메모리 오버헤드가 큼.
- TFHE(Torus Fully Homomorphic Encryption): 비트 단위로 데이터를 암호화하고, 논리 연산(AND, OR 등)을 효율적으로 수행할 수 있는 동형암호 방식.
  - programmable TFHE 이용 시 기존 TFHE보다 넓은 도메인 구간 처리가 가능하지만 여전히 계산 오버헤드와 메모리 사용량이 큼.

⇒ 넓은 범위의 평가는 가능하지만 연산 시 비용과 메모리 오버헤드 문제가 여전히 발생함.

Contribution < Domain-Extension Methodology >

❖ Domain-Extension Methodology

- **배경:** 입력 범위를 더 넓게 설정하면서도 적은 곱셈 연산으로 다항식으로 근사할 수 있는 기법의 필요성
- **목표:** **낮은 차수의 다항식**을 사용하면서도 **넓은 구간**에서 안정적인 근사 수행
- **구간 구분**
  - **Interval type I :** 함수가 **원래 다항식과 유사**하게 동작하는 구간, **입력값의 대부분**이 속하는 구간.
  - **Interval type II :** Interval type I 외부의 **확장된 구간**으로, **입력값이 크더라도** 함수 값이 안정적으로 유지되도록 설계된 구간.
- **방법:** Interval type II를 확장하여 큰 입력 값 처리 시에도 함수 값이 안정적으로 제한되도록 함.

  → 다항식이 정의된 도메인을 비율 L만큼 확장: $[-R, R] \rightarrow [-LR, LR]$

  → **원래 함수의 특성**을 유지하면서 넓은 범위에서도 안정적인 근사가 가능하도록 설계됨.
- 각 도메인 확장 함수($\boldsymbol{DEF}$: $Domain\ Extension\ Functions$) 와 도메인 확장 다항식($\boldsymbol{DEP}$: $Domain\ Extension\ Polynomials$) 을 **반복적으로 적용**하는 방식으로 동작.

DEFs (Domain Extension Functions)

❖ DEF (Domain Extension Functions)

- **주어진 함수의 입력 범위를 확장하면서도 기존 도메인에서의 함수 특성을 유지하는 역할**

$$\boldsymbol{D(x) = \frac{1}{2}(|x+1| - |x-1|)}$$

- 기존 함수 $P(x)$가 도메인 $[-1, 1]$에서 정의되어 있을 때, DEF 적용 시 확장된 도메인 $[-L, L]$에서도 $P(x)$를 사용할 수 있음.

⇒ 즉, **기존 함수 $P(x)$ 의 성질을 유지**(Interval type I: $[-1, 1]$)하면서 **입력 범위를 넓게 확장**(Interval type II: $[-L, L]$)할 수 있도록 해줌

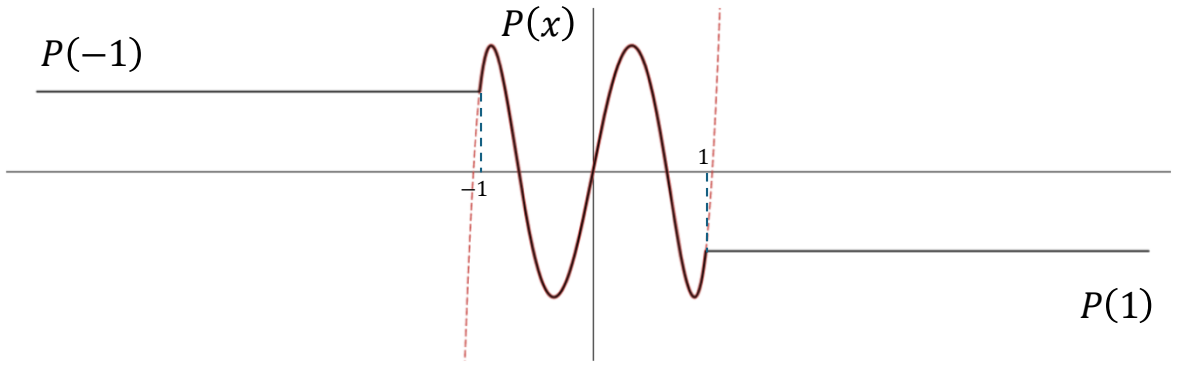
$$P \circ D(x) = \begin{cases} P(-1), & if \quad -L \le x \le -1 \\ P(x), & if \quad -1 \le x \le 1 \\ P(1), & if \quad 1 \le x \le L \end{cases}$$

⇒ DEF를 n번 반복 적용하여 $[-1, 1] \rightarrow [-L^n, L^n]$ 으로 확장하여 지수적으로 근사 구간 확대 가능

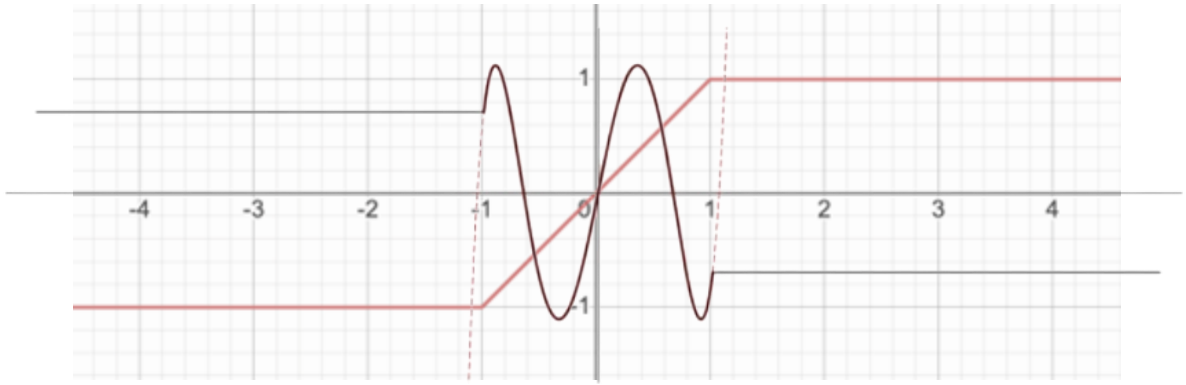
DEFs (Domain Extension Functions)

❖ DEF (Domain Extension Functions)

- **기존 함수 $P(x)$ 의 성질을 유지**(Interval type I: $[-1,1]$)하면서 **입력 범위를 넓게 확장**(Interval type II: $[-L,L]$)
  - $[-1,1]$ 에서는 단순히 항등 함수처럼 작동해 원래 함수의 특성을 정확히 보존하며, 이를 초과하는 범위(Interval type II)에 대해서는 $P(1), P(-1)$ 값으로 고정하여 암호문 연산이 가능한 공간 안에 있도록 유지함으로써 HE 연산 실패를 방지함.



[그림 1] DEF 예시

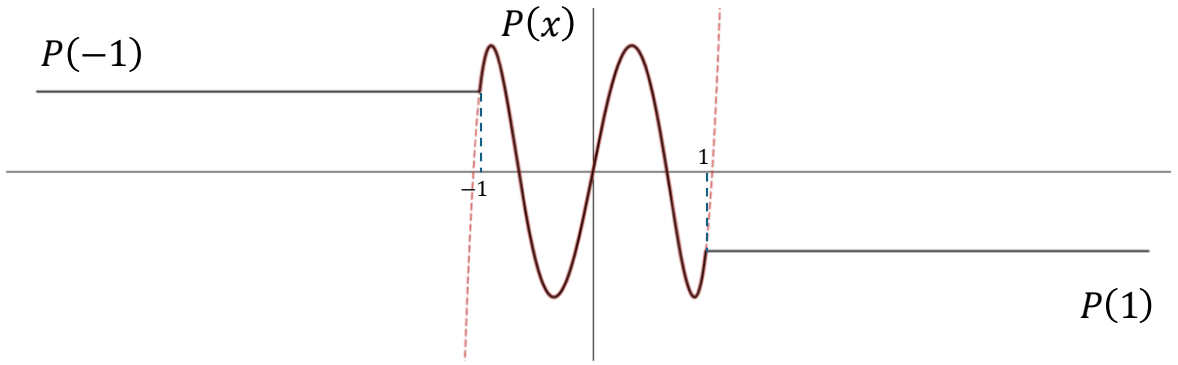


[그림 2] D(x) 함수 참고

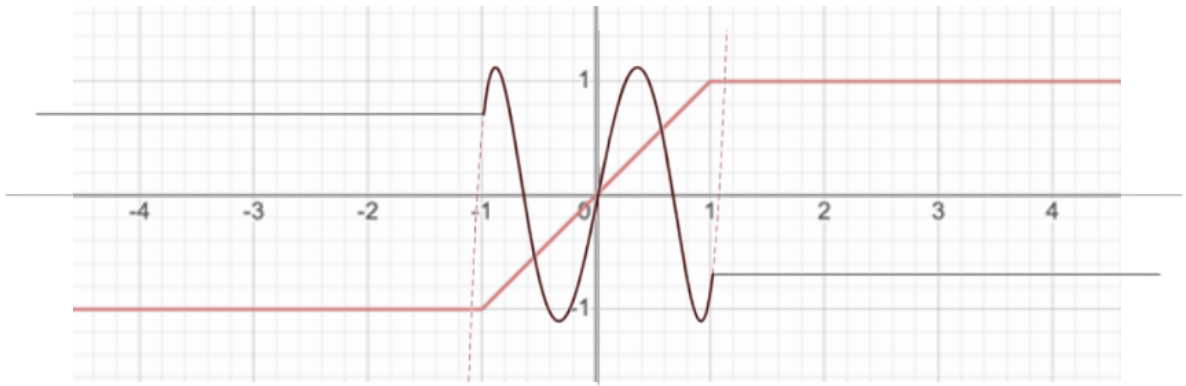
DEFs (Domain Extension Functions)

❖ DEF (Domain Extension Functions)

- **기존 함수 $P(x)$ 의 성질을 유지**(Interval type I: $[-1,1]$)하면서 **입력 범위를 넓게 확장**(Interval type II: $[-L,L]$)
  - $[-1,1]$ 에서는 단순히 항등 함수처럼 작동해 원래 함수의 특성을 정확히 보존하며, 이를 초과하는 범위(Interval type II)에 대해서는 $P(1), P(-1)$ 값으로 고정하여 암호문 연산이 가능한 공간 안에 있도록 유지함으로써 HE 연산 실패를 방지함.



[그림 1] DEF 예시



[그림 2] D(x) 함수 참고

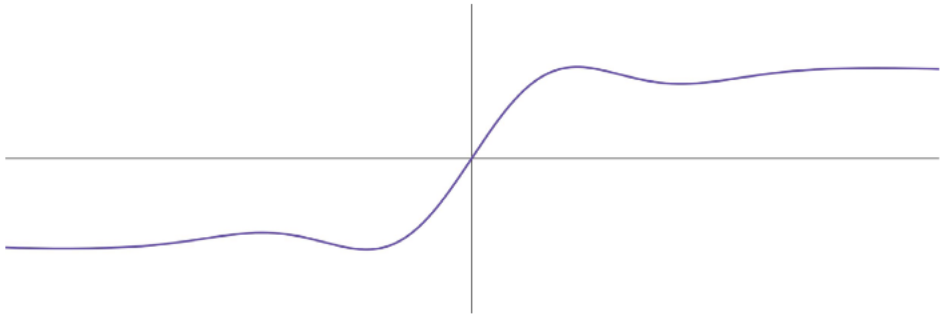
➜ 그러나 HE 환경에서는 비선형 함수를 연산할 수 없어 DEF 를 직접 적용할 수 없기 때문에, 이를 대체할 다항식 형태로 함수를 정의해야 함.

DEPs (Domain Extension Polynomials)

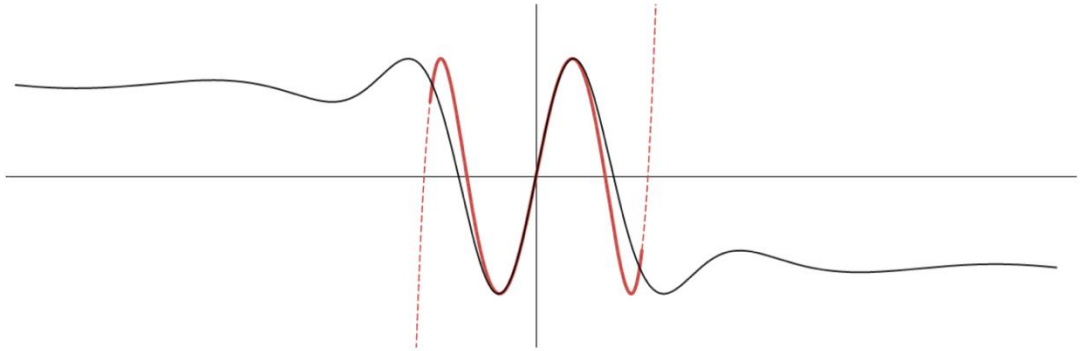
❖ DEP (Domain Extension Polynomials)

- **DEF를 다항식으로 근사**하여 **HE 환경**에서도 직접적으로 활용할 수 있도록 한 방법.
- DEP의 정의: $d(x) \in D(\delta, r, R_1, R_2)$ 는 다음 조건을 만족하는 다항식 (δ : 근사오차 상한, r 및 R_1: Interval type I,II, R_2: R_1에 확장 비율 L 적용)
  - Interval type I 구간의 근사 정확성: $|x - d(x)| \leq \delta |x|^3 \quad \forall x \in [-r, r]$
  - Interval type I 구간의 기울기 제한: $0 \leq d'(x) \leq 1 \quad \forall x \in [-r, r]$

  ⇒ Interval type II를 $[-R_1, R_1]$에서 $[-R_2, R_2]$로 확장하면서, Interval type I [−r, r] 에서 원래 함수의 특징을 보존



[그림 3] DEP 예시



[그림 4] DEP를 이용한 도메인 확장 예시

Domain-Extension Methodology With DEPs

❖ DEP (Domain Extension Polynomials) Algorithm 1

- **DEP를 반복적으로 적용**하여 작은 도메인에서 정의된 다항식을 큰 도메인으로 **점진적으로 확장** 가능
- 확장 과정에서 중앙 부분(interval type Ⅰ)은 고정되며, 외부 영역(interval type Ⅱ)은 확장됨.
  - 이 과정에서 **원래 다항식의 특징이 유지**되며, **넓은 구간에서도 유사한 근사** 성능을 제공할 수 있도록 함.

Algorithm 1 Homomorphic Evaluation of $C_{\lim}$ Functions on Large Intervals

Input: $f(\cdot) \in C_{\lim}, x \in [-L^n R, L^n R]$. → 무한대로 수렴하는 함수 $f(x)$와 확장된 입력 구간을 Input으로 받음

Output: Approximate value of $f(x)$.

1: $P(\cdot) :=$ the minimax polynomial on $[-R, R]$ → 함수 $f(x)$를 $[-R, R]$에서 Minimax 다항식 $P(x)$로 근사

2: Take $B(\cdot) \leftarrow \mathscr{D}(\delta, r, R, LR)$ → 도메인 확장 다항식(DEP) $B(x)$ 선택

3: $y = x$

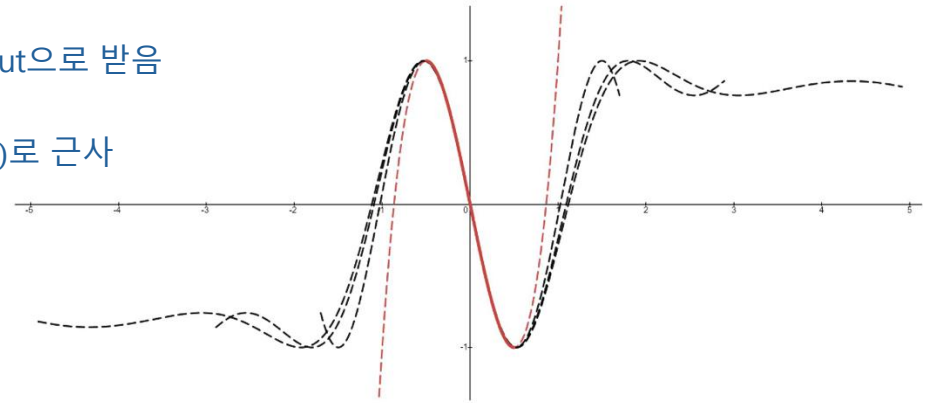
4: **for** $i \leftarrow n-1 \rightarrow 0$ **do**

5: $y = L^i B\left(\frac{y}{L^i}\right)$ → DEP($B(x)$) 를 n번 반복 적용

6: **end for**

7: $y = P(y)$ → DEP를 적용한 값에 미니맥스 다항식 근사로 정확한 함수의 근사값 계산

8: **return** y



[그림 5] Domain Extension 방법을 3회 적용한 후 확장된 interval type II 에서의 함수들로, [-0.5, 0.5] 에서 원래 함수의 특성이 유지되는 예시

Minimax Polynomial Approximation

❖ Minimax Polynomial Approximation

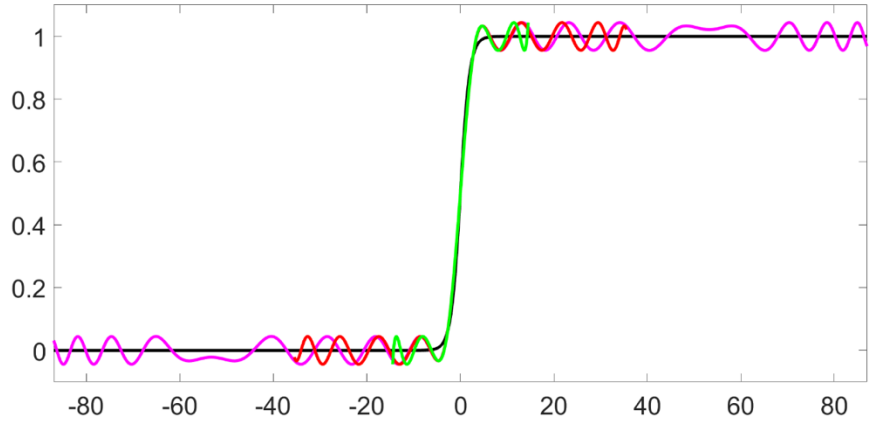
- 주어진 구간에서 **연속 함수 $f(x)$를 근사하는 차수가 최대 d인 다항식 $p(x)$를 찾아 최대 오차를 최소화**하는 방법
  - 도메인 구간 $[-R, R]$ 에서 최대 오차가 고정된 근사 연산 시 Minimax 근사를 평가하는 과정에서 곱셈 연산을 줄이기 위해 $Paterson - Stockmeyer$ 알고리즘을 이용하여 $\boldsymbol{O(\sqrt{d})}$번의 곱셈 수행
    - **Paterson-Stockmeyer 알고리즘**: 고차 다항식을 효율적으로 계산하는 알고리즘

    → 차수가 d인 다항식을 평가할 때 $\boldsymbol{O}(\sqrt{\boldsymbol{d}})$개의 곱셈만 사용할 수 있도록 최적화

Domain-Extension Methodology With DEPs

❖ Uniform Approximation Using DEPs

- 무한대에서 수렴하는 함수 $f(\cdot) \in C_{\lim}$에 대한 효율적인 다항식 근사 기법
  - 기계 학습 알고리즘은 무한대에서 수렴하는 함수(logistic, tanh 함수 등) 사용
    ⇒ DEP 사용 시 넓은 입력 구간에서도 정확하고 안정적인 근사 가능



[그림 6] 두 번의 도메인 확장을 통한 Logistic 함수 근사 과정

- Logistic 함수에 DEP을 적용해 도메인 확장 횟수에 따른 곱셈 수와 최대 오차 변화를 살펴봄.
  - 확장 범위: $[-14.5, 14.5]$를 2.45배씩 3번 증가시킨 범위 → $[-32.525, 32.525]$

[표 1] 도메인 확장에 따른 Logistic 함수 근사의 오차 및 곱셈 수

# of extensions	0	1	2	3
Size of domain	29	71.05	174.07	426.48
Maximum error	0.04416	0.04447	0.04447	0.04447
# of Mult	4	6	8	10

DEPs with Higher Accuracy

❖ DEP (Domain Extension Polynomials) Algorithm 2

- 기존 방법에서는 함수 $f(x)$를 직접 minimax 다항식 $P(x)$로 근사하였으나, 도메인 확장 과정에서 $f \circ d^{-1}$ 를 근사하는 것이 더 정확한 결과를 제공할 수 있음.
- 이 과정에서 5차 다항식을 추가로 사용하여 정확도를 높임 → ***Max Error*** **를** 2^{-20} **이하로 확보** 가능

Algorithm 2 Homomorphic Evaluation of C_{lim} Functions on Large Intervals With Higher Accuracy

Input: $f(\cdot) \in C_{\text{lim}}, x \in [-L^n R, L^n R]$.
Output: Approximate value of $f(x)$.

1: $P(\cdot) :=$ the minimax polynomial on $[-R, R]$
2: $y = x$
3: **for** $i \leftarrow n-1 \rightarrow 0$ **do**
4: $\quad y = y - \frac{4}{R^2 27 L^{2i}} y^3$
5: **end for**
6: $y = y/R$
7: $y = y + \frac{4}{27} \frac{L^2(L^{2n}-1)}{L^{2n}(L^2-1)}(y^3 - y^5)$
8: $y = P(Ry)$
9: **return** y

[표 2] DEP 의 반복 적용된 예시

Maximum Error ($\log_2$)					
# of extensions (Size of domain)	0 (110)	1 (220)	2 (440)	3 (880)	4 (1760)
Algorithm 1	-21.6	-14.0	-13.7	-13.6	-13.6
Algorithm 2	-21.6	-20.3	-20.1	-20.0	-20.0

- 확장 범위: $[-55, 55]$를 2배씩 4번 증가시킨 범위 → $[-1760, 1760]$

Computational & Memory Cost

❖ Comparing Computational & Memory Cost with Related Works

- **[14] HE 스킴 간 변환(Pegasus)**: TFHE와 CKKS 변환을 활용하여 비다항식 함수 연산을 가능하게 하는 방법.

  ⇒ TFHE 이용 시 SIMD(Single Instruction, Multiple Data) 연산 활용 불가 → 병렬 처리가 어렵기 때문에 실행 시간이 $\boldsymbol{O(R)}$에 비례.

- **Minimax polynomial approximation**: 주어진 최대 오차 하에서 최소 차수로의 다항식 근사 후 평가 시 Paterson-Stockmeyer 알고리즘으로 곱셈 연산 횟수 줄임. ⇒ 도메인 $[-R, R]$ 평가 시 다항식 차수 $\Omega(R)$ 필요.

- **Domain-Extension Methodology With DEP**: 넓은 도메인 $[-R, R]$을 바로 근사하지 않고 작은 도메인 $[-r, r]$에서 **낮은 차수의 다항식으로 먼저 근사 후 이를 반복 적용**함으로써 $\boldsymbol{O(log\, R)}$의 낮은 곱셈 횟수 달성.

  - **이전 반복의 중간 결과를 저장하지 않아 추가적인 메모리 소비 없이 단일 값을 업데이트** 할 수 있어 $\boldsymbol{O(1)}$ 메모리 사용 가능.

[표 3] $[-R, R]$ 구간에서의 계산 비용 및 메모리 사용 비교

	# of Mult	# of cMult	Mult. depth	Memory
[14]	Time: $\Omega(R)$			$\Omega(R)$
Minimax	$\Omega(\sqrt{R})$	$\Omega(R)$	$\Omega(\log R)$	$\Omega(\sqrt{R})$
Ours	$O(\log R)$	$O(\log R)$	$O(\log R)$	$O(1)$

Experiments

Experiments

❖ Experimental Objectives and Overview

- 도메인 확장 방법론(DEP)의 성능 검증 목적으로 프라이버시 보호가 필요한 환경에서 로지스틱 회귀 모델의 활용 가능성을 확인하고, 기존 방법론에 대한 개선점을 제시함.

- **실험1) 넓은 구간에서의 로지스틱 함수 근사 성능 평가**
  - 도메인 확장 기법을 적용하여 기존 방식과 비교했을 때 얼마나 더 효율적으로 로지스틱 함수를 근사할 수 있는지 확인.
  - **최대 오차 ≈ 0.045** (중간 정확도) 와 **최대 오차 $\leq 2^{-20}$** (높은 정확도) 수준에서 평가.
    ✓ Minimax vs. DEP Algorithm 1 / Pegasus vs. DEP Algorithm 1 / Minimax vs. Algorithm 1,2

- **실험2) 동형암호 기반 프라이버시 보호 로지스틱 회귀 학습 및 추론**
  - 동형암호 환경에서 프라이버시를 보장하면서도 Logistic Regression 모델을 잘 학습할 수 있는지 검증.
  - MNIST(입력 범위: $[0, 255]^{28\times28}$) , Swarm Behavior(명시적 입력 상한 없음) 데이터셋을 활용하여 실험 수행.

Homomorphic Evaluation of the Logistic Function

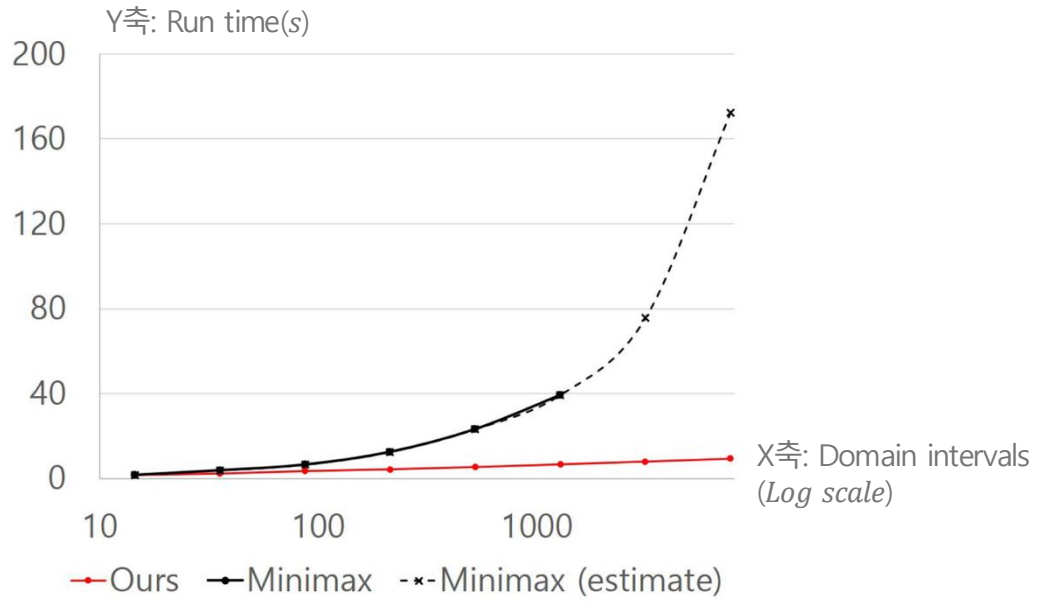
❖ 1) Minimax vs. DEP Algorithm 1

- 로지스틱 함수 근사에 9차 Minimax 다항식 사용, **최대 오차 ≈ 0.045 (중간 정확도) 수준**에서 평가
  - 도메인 구간: $[-14.5\sim14.5] \sim [-7683, 7683]$ (각 단계에서 2.45배씩 증가) / 사용된 DEP: $B(x) = x - \frac{4}{27}\frac{1}{14.5^2}x^3$

⇒ 도메인 크기가 커짐에 따라 DEP가 더 효율적($O(log\ R)$)

[표 4] Minimax vs. DEP Algorithm 1

Size of domain	Minimax		Algorithm 1	
	Max error	Runtime (sec)	Max error	Runtime (sec)
29	0.04416	1.57	0.04416	1.57
71	0.04161	3.72	0.04441	2.30
174	0.04389	6.42	0.04445	3.51
426	0.04376	12.43	0.04446	4.33
1045	0.04477	23.18	0.04446	5.44
2560	0.04479	39.28	0.04446	6.72
6272	-	(75.54)	0.04446	8.04
15366	-	(172.46)	0.04446	9.43



[그림 7] Minimax vs. Algoritm1

Homomorphic Evaluation of the Logistic Function

❖ 2) Pegasus vs. DEP Algorithm 1

- Pegasus를 사용하여 큰 도메인 구간 $[-R, R]$에서 함수 $f(\cdot)$를 평가하기 위해, $f(Rx/8)$을 $[-8,8]$ 구간에서 평가
  - Pegasus: CKKS 암호문을 TFHE 형태로 변환한 후, Look-up Table (LUT)을 사용하여 값을 참조
- Amortized time: Pegasus에서 SIMD 연산이 불가능하기 때문에, 각 연산을 개별적으로 처리(CKKS 슬롯들을 각각 TFHE로 변환)해야 함. 이때 개별 입력마다 처리 시간의 편차가 크기 때문에 여러 입력을 평가 후 평균 시간 산출.

[표 5] Pegasus vs. DEP Algorithm 1

Size of domain	Pegasus [14]		Algorithm 1	
	Max error	Amortized time (ms)	Max error	Amortized time (ms)
29	0.01470	837.01	0.04416	0.048
71	0.02617		0.04441	0.070
174	0.04968		0.04445	0.107
426	0.12189		0.04446	0.132
1045	0.33802		0.04446	0.166
2560	0.73180		0.04446	0.205
6272	0.96092		0.04446	0.245
15366	0.97499		0.04446	0.288

✓ **29~71 범위에서 Pegasus의 더 낮은 오차**
Pegasus의 LUT 방식은 미리 계산된 함수 값을 참조하는 방식이므로, 작은 도메인에서는 **LUT에서 보다 정확한 값**을 불러올 수 있어 상대적으로 낮은 오차가 발생.

✓ **나머지 범위에서의 Pegasus의 높은 오차**
LUT는 일반적으로 **특정 범위의 입력값**에 대해서만 정의되기 때문에, 만약 입력값이 해당 범위를 벗어난다면, LUT에서 직접적으로 적절한 값을 찾을 수 없고, 그에 따라 **가장 근접한 결과값으로 대체**할 때 오차가 발생.

✓ **DEP 방식의 비교적 빠른 실행 시간**
CKKS에 기반한 SIMD 적용으로 한 번의 연산으로 실수값을 동시에 평가할 수 있어 평균 실행 시간 측정 시 빠른 속도를 보임.

Homomorphic Evaluation of the Logistic Function

❖ 3) Minimax vs. Algorithm 1,2

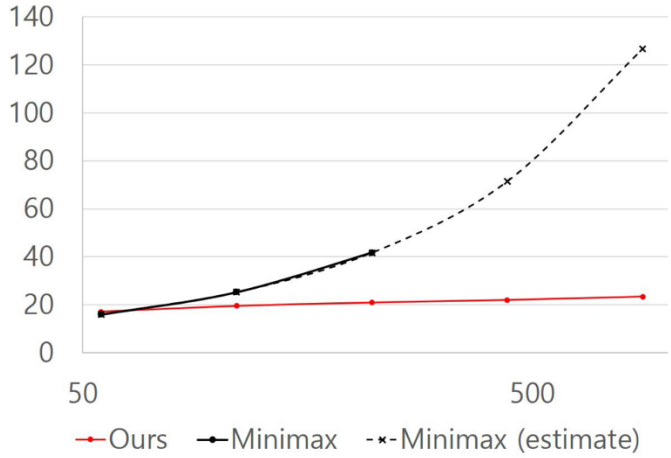
- Algorithm2: 다항식 근사의 **최대 오차 $\leq 2^{-20}$ (높은 정확도)** 수준에서 평가
- 사용된 DEP: $B(x) = x - \frac{4}{27}\frac{1}{55^2}x^3$, Minimax 방법을 통해 얻은 243차의 $P(x)$를 기반으로 수행
- 도메인 구간: $[-55{\sim}55] \sim [-880, 880]$ (각 단계에서 2배씩 증가)

✓ **DEP Algorithm 2**
더 높은 정확도를 제공하기 위한 추가적인 연산을 수행하므로 Algorithm 1에 비해 조금 더 높은 실행시간을 보임.

✓ **DEP 방식의 비교적 빠른 실행 시간**
넓은 범위까지 연산 시 높은 차수의 다항식을 직접 계산하지 않고, 낮은 차수 다항식 여러 개를 계산하여 도메인을 확장하기 때문. ($O(\log R)$)

[표 6] Minimax vs. Algorithm 1,2

Size of Domain	Minimax		Algorithm 1		Algorithm 2	
	Error ($\log_2$)	Time (sec)	Error ($\log_2$)	Time (sec)	Error ($\log_2$)	Time (sec)
110	-20.0	16.0	-21.6	17.1	**-21.6**	17.2
220	-20.1	25.2	-14.0	17.7	**-20.3**	19.5
440	-20.0	41.7	-13.7	18.8	**-20.1**	20.9
880	-	(71.4)	-13.6	20.0	**-20.0**	22.0
1760	-	(126.7)	-13.6	21.3	**-20.0**	23.4



[그림 8] Minimax vs. Algoritm2

Experiment using MNIST

❖ MNIST 데이터셋을 활용한 CKKS 기반 Logistic Regression 실험

- DEP: $B(x) = x - \frac{4}{27}x^3$, Logistic 함수의 15차 minimax 다항식인 $P(x)$의 $[-\mathbf{14.5}, \mathbf{14.5}]$ **구간에서 확장 비율 2.45로** $[-\mathbf{7683}, \mathbf{7683}]$**까지 확장**
- MNIST Train 데이터: 60,000개 중 9,600개 선택 / Test 데이터: 10,000개 중 3또는 8 라벨의 샘플 선택

  ⇒ **모델 학습 과정에서 다항식 근사가 지속적으로 유효한지 확인하기 위한 목적**

- **Logistic Regression 학습 방식**
  - 확률적 경사 하강법 사용, mini-batch: 320, **학습 반복 횟수: 30회(1 epoch)**, 학습률: 1.0, 0.1
  - Logistic Regression에서는 모델이 학습될수록 가중치 벡터 w_t가 업데이트됨에 따라 입력값$(x = w_t \cdot x_t)$도 변함.
- 반복(iteration)이 증가할수록 가중치가 학습됨에 따라 가중치 벡터 w_t의 크기를 계산해 Logistic 함수의 입력 범위를 결정함.

⇒ **1 epoch (총 학습 반복횟수인 30회) 동안의 정확도 변화를 측정**

⇒ **각 반복에서 입력 값의 최대 크기를 모니터링하여 학습 과정에서도 도메인 확장 기법을 적용한 다항식 근사가 유효한지 평가**

Experiment using MNIST - Results

❖ MNIST 데이터셋을 활용한 CKKS 기반 Logistic Regression 실험 결과

- 학습률에 따른 정확도 및 최대 입력값의 크기 변화를 측정
  - 학습률 0.1 : 안정적이지만 학습 속도가 상대적으로 느림
  - 학습률 1.0 : 모델이 더 빠르게 수렴하고, 최종적으로 더 높은 정확도(96.11%) 달성

    → w_t의 크기가 빠르게 증가하면서, 학습 도중 Logistic 함수의 입력 값도 커짐.

[표 7] HE 기반 MNIST 데이터셋 Logistic Regression 학습 결과

# of iterations	Accuracy (0.1)	Max input (0.1)	Accuracy (1.0)	Max input (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

✓ **학습률 1.0의 큰 Max input**
학습률이 클수록(1.0) 가중치의 변화폭이 커져 입력값의 범위도 작은 학습률(0.1)일 때에 비해 비교적 큰 입력값을 보임.

✓ **학습률 1.0의 50.9%의 정확도**
50.9%의 정확도는 거의 랜덤 추측 수준에 가까운데, 이는 가중치가 학습 초기에 불안정하게 변화하여 모델이 아직 의미 있는 학습을 하지 못했기 때문이라 추측.
(초기에는 불안정한 값을 가짐)

✓ **학습률 1.0 의 입력값 수렴**
학습이 진행될수록 입력 값이 안정되어 최대 입력 값이 약 10~15 사이에서 수렴하는 경향을 보임.

Experiment using Swarm Behavior Dataset

❖ Swarm Behavior 데이터셋을 활용한 CKKS 기반 Logistic Regression 실험

- Swarm Behavior Dataset을 사용하여 Logistic Regression 모델을 학습해 'aligned' 태그에 대해 이진 분류 수행
- Train 데이터: 24,016개 중 3,840개 샘플 선택 / Test 데이터: 나머지 샘플
- 2400개의 feature을 가지는 고차원 데이터셋으로, HE 환경에서 처리하기 위해서는 적절한 도메인 확장 및 다항식 근사가 필요.

⇒ **상대 오차(relative error)**를 분석하여 **암호화된 상태(HE)와 일반적인 평문 상태(plaintext)**에서 학습한 모델의 성능을 비교

- **Logistic Regression 학습 방식**
  - 확률적 경사 하강법 사용, mini-batch: 240, **학습 반복 횟수: 16회(1 epoch)**, 학습률: 10^{-6}

Experiment using Swarm Behavior Dataset - Results

❖ Swarm Behavior 데이터셋을 활용한 CKKS 기반 Logistic Regression 실험 결과

- 암호화된 모델과 일반 모델 간 정확도 차이가 작음 ⇒ **HE 환경에서도 정확도가 평문 모델과 거의 동일하게 유지**됨.

[표 8] HE 기반 Swarm Behavior 데이터셋 Logistic Regression 학습 결과

# of iterations	Accuracy (ciphertext)	Accuracy (plaintext)	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

✓ **Max input** : Swarm Behavior 데이터셋은 입력 벡터 크기가 일정하지 않고, 특정 반복에서 갑자기 큰 값이 등장할 수 있음.

✓ **Relative error**
암호화된 모델은 학습이 진행될수록 연산을 반복함에 따라 암호문 내에서 오차가 누적된다. 따라서 이러한 이유 때문에 초기에는 CKKS 연산의 근사 오차가 작지만, 반복이 거듭될수록 오차가 점점 누적되면서 상대 오차가 증가한다고 생각.

✓ **Relative Error가 증가함에도 불구하고 정확도 차이가 크지 않은 이유**
CKKS 기반의 근사 연산에서 생기는 작은 오차들이 존재하지만, 가중치가 다소 변하더라도 Logistic Regression를 이용한 학습 과정에서 출력값이 0 또는 1에 수렴하기 때문에 오차를 보정하는 역할을 하여 최종 분류 성능이 유지될 수 있다고 생각.

Conclusion

❖ 결론

- **로지스틱 회귀 모델과 같은 머신러닝 모델에서 넓은 도메인의 대규모 데이터셋에서도 동형암호 기반 연산이 가능하도록 방법을 제안.**

  ⇒ 낮은 차수의 다항식을 반복적 적용하여 확장하는 **DEP(Domain-Extension Methodology) 기법**을 활용하여 **넓은 구간**에서도 높은 정확도로 함수 평가가 가능함.

- $O(\log R)$ 연산량과 $O(1)$ 메모리만 사용하여, 기존 연구의 minimax 다항식 근사를 이용한 $O(\sqrt{R})$ 연산량과 $O(\sqrt{R})$ 메모리를 요구하는 방식보다 효율적으로 개선할 수 있었음.
- 향후 연구에서는 임의의 함수에 대한 효율적인 동형암호 기반 평가 방법을 연구해볼 수 있음.

End