Research Article

Privacy-Preserving Outsourced Logistic Regression on Encrypted Data from Homomorphic Encryption

Xiaopeng Yu,¹ Wei Zhao,¹ Yunfan Huang,¹ Juan Ren,¹ and Dianhua Tang,¹,²

¹Science and Technology on Communication Security Laboratory, Chengdu 610041, China
²School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Correspondence should be addressed to Dianhua Tang; tangdianhua86@163.com

Received 23 March 2022; Revised 11 May 2022; Accepted 24 May 2022; Published 21 July 2022

Academic Editor: Debiao He

Copyright © 2022 Xiaopeng Yu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Logistic regression is a data statistical technique, which is used to predict the probability that an event occurs. For some scenarios where the storage capabilities and computing resources of the data owner are limited, the data owner wants to train the logistic regression model on the cloud service provider, while the high sensitivity of training data requires effective privacy protection methods that enable efficient model training without exposing information about the training data to untrusted cloud service providers. Recently, several works have used cryptographic techniques to implement privacy-preserving logistic regression in such application scenarios. However, on large-scale training datasets, the existing works still have the problems of long model training time and poor model performance. To solve these problems, based on the homomorphic encryption (HE), we propose an efficient privacy-preserving outsourced logistic regression (P²OLR) on encrypted training data, which enables data owners to utilize the powerful storage and computing resources of cloud service providers for logistic regression analysis without exposing data privacy. Furthermore, the proposed scheme can pack multiple messages into one ciphertext and perform the same arithmetic evaluations on multiple plaintext slots by using the batching technique and single instruction multiple data (SIMD) mechanism in HE. On three public training datasets, the experimental results show that, compared with the existing schemes, the proposed scheme has better performance in terms of the encryption and decryption time of the data owner, the storage of encrypted training data, and the training time and accuracy of the model.

1. Introduction

Logistic regression (LR) [1] is a popular classification method, which has been used in numerous practical applications including cancer diagnosis [2], credit scoring [3], genome-wide association study [4], and more. LR can not only be applied to the problem of predicting the probability of occurrence of various events, but also is competitive with other classification algorithms in terms of prediction accuracy. In some practical application setting, the data owners have the limited computing and storage resources, and thus wants to outsource some of the heavy computation in logistic regression model training, the outsourced data analysis [5] has received considerable attention recently, which enables data owners to train a LR model using the powerful storage capacity and computing resources of cloud service providers [6]. However, the high sensitivity of training data requires to perform an effective privacy protection [7–10] that enable efficient and secure logistic regression analysis without leaking information about the training data to untrusted cloud service provider. Recently, to meet such application requirements, based on the cryptographic techniques like secure multiparty computation (MPC) [11] and homomorphic encryption (HE) [12], there have been several researches on the privacy-preserving logistic regression (PPLR) [13–22], which enables data owners to employ the service providers’ powerful data storage and computing resources for logistic regression model training without exposing its own data privacy. Specifically, the data owner encrypts its training data, and sends encrypted training data to the service provider. The service provider can train a logistic regression model on encrypted training data, and
returns the encrypted training result to the data owner. The data owner can decrypt the encrypted training result to obtain final training result.

Unfortunately, on large-scale training dataset, the existing PPLR schemes [13–22] still have the bottlenecks of high model training time and low model precision. To solve these problems, based on the HE cryptographic technique [23] that has the property that the operation results on ciphertexts are consistent with those on plaintexts, we design an efficient privacy-preserving outsourced logistic regression (P^2OLR). The main contributions are as follows:

(1) Firstly, we propose a method for achieving P^2OLR on encrypted data from HE. To speed up the model training, the proposed P^2OLR scheme employs the batching technique to pack multiple elements into multiple plaintext slots, encrypts them into one ciphertext, and performs the same arithmetic operations to multiple plaintext slots in the SIMD mechanism.

(2) Secondly, we evaluate the proposed P^2OLR on three public datasets [18]. Under the same experimental environment, compared with the related P^2OLR [17, 18, 22], the model training time of the proposed P^2OLR is reduced by more than 71.7%, and the proposed P^2OLR has a better model performance.

The rest of this paper is arranged as follows. We present the related works in Section 2. We review the preliminaries related to our P^2OLR in Section 3. In Section 4, our P^2OLR is described. The performance evaluation for our P^2OLR is presented in Section 5. The security analysis of our P^2OLR is shown in Section 6. Finally, we conclude in Section 7.

2. Related Works

There have been a lot of works on achieving PPLR using cryptographic techniques. In this paper, we mainly focus on the PPLR based on HE. To outsource the LR model training to a cloud service provider in a privacy-preserving manner, based on the HE scheme (FV) [24], Charlotte et al. [13] proposed an algorithm to train a LR model on an homomorphically encrypted dataset, which is implemented based on the FV-NFlib library [25]. However, the accuracy of model is poor due to the use of a quadratic polynomial to approximate the sigmoid function. Furthermore, the training time grows linearly in the number of training samples. Using the HE scheme (FV) [24] and 1 bit gradient descent (GD) method, Chen et al. [14] presented a method to train LR over encrypted data, which is implemented through the SEAL library [26], and allows an arbitrary number of iterations by using bootstrapping [27] in FV, but bootstrapping introduces a significant decrease in performance. Focusing on the prediction process of LR, based on the HE scheme (BGV) [28], Li and Sun [15] proposed a secure protocol to solve the data leakage problem during the LR prediction process, and implement their scheme by the HElib library [29]. Based on the Chimera framework [30] that allows switching between HE schemes TFHE [31] and CKKS [23], Carpov et al. [16] proposed a solution to achieve semi-parallel LR on encrypted genomic data, which performs the bootstrapping [27] without re-encrypting the genomic data for an arbitrary number of iterations, and is implemented by using TFHE library [32] and HEAAN library [33].

Adapting the packing and parallelization techniques of approximate HE scheme (CKKS) [23], Kim et al. [17] proposed a PPLR, which is implemented through using the HEAAN library [33], and uses least squares approximation to improve the accuracy and efficiency of LR model training. However, as the number of iterations increases, the parameters of the CKKS scheme also need to become larger, which makes the training time increase dramatically. Kim et al. [18] applied the HE scheme (CKKS) [23] to achieve PPLR. Their scheme is implemented via using the HEAAN library [33]. Moreover, they devised an encoding method to decrease the storage of encrypted training data and adapted Nesterov’s accelerated GD method to reduce the number of iterations as well as the computational cost. However, their scheme requires the assumption that both the number of training samples and features are power-of-two, which makes the scheme unsuitable for practical applications. To reduce the number of iterations, Cheon et al. [19] proposed an ensemble GD method based on the HE scheme (CKKS) [23], and applied it to the PPLR, in which they approximate the sigmoid function using a polynomial of 5-degree obtained by least squares approximation. Their scheme is implemented based on the HEAAN library [34]. To run a genome-wide association study on encrypted data, using the SIMD capabilities of HE scheme (CKKS) and Nesterov’s accelerated GD, Bergamaschi et al. [20] introduced a method for homomorphic training of LR model, which is implemented based on the HElib library [29]. To protect the private information of both parties, based on the HE scheme (CKKS) [23] and gradient sharing technology, Wei et al. [21] proposed a protocol to train an LR model on vertically distributed data between two parties, which does not require trusted third-party nodes and is implemented by the HElib library [29]. Based on the HE scheme (CKKS) [23], Fan et al. [22] offered a PPLR algorithm, where they approximate the sigmoid function in LR by Taylor’s theorem, and use row encoding to encrypt training samples, but as the number of samples increased, this will lead to longer model training time.

3. Preliminaries

3.1. System Model. As can be seen in Figure 1, the system model of the proposed P^2OLR considers two entities, namely a data owner (DO) and a service provider (SP). For readability, the definitions of the notations in this paper are shown in Table 1. DO: It has limited computational resources, and wants to use SP’s data analysis service on encrypted data to train a LR model without revealing its own training data privacy. SP: It is a semi-trusted entity with powerful data storage and computing capabilities, and can provide data analysis and statistical services on encrypted data for DO. Specifically, DO chooses poly_modulus_degree N, coeff_modulus Q, and runs key_generation algorithm to
generate the secret_key sk, public_key pk, relinearization_key rk, galois_key gk. Next, DO encrypts the training data D ∈ ℜ^(m×n) into ciphertexts D, encrypts the initial weight {w_0^{(0)}, w_1^{(0)}, ..., w_n^{(0)}} into ciphertexts W^{(0)}, encrypts the learning rate α into one ciphertext α/m, and sends N, Q, Δ, pk, rk, gk, t, D, W^{(0)}, α/m to SP. SP performs the P²OLR algorithm and returns the ciphertext result W(t) of the t-th iteration to DO. DO decrypts the ciphertext result W(t) to obtain final result {w_0^{(f)}, w_1^{(f)}, ..., w_n^{(f)}}.

3.2. Homomorphic Encryption (HE). Homomorphic encryption (HE) is a cryptographic technique, which allows operations on ciphertexts without decryption, and guarantees that the computation results on ciphertexts are consistent with the computation results on plaintexts. We adopt the HE scheme (CKKS) [23] based on the Ring Learning with Errors (RLWE) problem, which can encrypt multiple elements in one ciphertext and supports the single instruction multiple data (SIMD) operations. Suppose ΦM(X) = X^N + 1 denotes the M-th cyclotomic polynomial, where N is power of 2. R_q = ℜ[X]/(X^N + 1) denotes the cyclotomic ring of polynomials. R_q = ℜ[X]/(X^N + 1) denotes the residue ring of R_q modulo q. ℍ denotes a subring of complex vector C^N that is isomorphic to ℜ/N. σ: ℜ → σ(ℜ)⊆ ℍ denotes a canonical embedding that transforms a plaintext polynomial R into a complex vector ℍ. π: ℍ → ℜ/N denotes a natural projection that transforms a complex vector C^N to ℜ/N. HE scheme (CKKS) [23] supports the operations as follows, which can be found in the Appendix. For ease of description, we define the Algorithms 1–9.

3.3. Sigmoid Approximation. Since the existing HE scheme can only effectively support polynomial arithmetic computations, the computation of sigmoid function σ(x) = 1/(1 + e^−x) using HE is a barrier to the realization of P²OLR. To find a approximate polynomial of σ(x), adapting the least squares method, we consider the 7-degree polynomial g(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 over the domain [−8, 8], where a_0 = 1/2, a_1 = 1.73496/8, a_2 = 4.19407/8, a_3 = 5.43402/8, a_4 = 2.50739/8. σ(x) and g(x) can be seen in Figure 2, the maximum errors between σ(x) and g(x) are about 0.032. g(x) over encrypted data x from HE can be achieved by the Algorithm 10.

3.4. Logistic Regression. Logistic regression (LR) is a statistical analysis method for predicting the probability of an event. We consider the case where the predicted value is a binary dependent variable. Assuming that a dataset consists of m samples of the form {y_i, x_i} with y_i ∈ {0, 1} and x_i = [x_{i1}, x_{i2}, ..., x_{iN/2−1}] ∈ ℜ^(n/2−1), the goal of LR is to find the optimal parameters w = [w_0, w_1, ..., w_n−1] that minimizes the negative log-likelihood (loss function) J(w) with respect to w is calculated by

\[
J(w) = \frac{1}{m} \cdot \sum_{i=0}^{m} \left( y_i \cdot \log(\sigma(d_i \cdot w^T)) + (1 - y_i) \cdot (1 - \log(\sigma(d_i \cdot w^T))) \right),
\]

for each vector d_i = [1, x_i]. A common method for minimizing loss function J(w) is a gradient descent (GD) algorithm, which finds the local extremum of a loss function by following the direction of the gradient. The gradient of J(w) is calculated by

\[
\frac{\partial J(w)}{\partial w} = \frac{1}{m} \cdot \sum_{i=0}^{m} (y_i - \sigma(d_i \cdot w^T)) \cdot d_i.
\]
### Table 1: The definitions of the notations.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>A message vector ([x_0, x_1, \ldots, x_{N/2-1}])</td>
</tr>
<tr>
<td>(\langle x \rangle)</td>
<td>The plaintext of message vector (x)</td>
</tr>
<tr>
<td>(X)</td>
<td>A list ([x_0, x_1, \ldots, x_{N-1}])</td>
</tr>
<tr>
<td>(X_i)</td>
<td>The ciphertext of ciphertext list (X_i)</td>
</tr>
<tr>
<td>(x \cdot y)</td>
<td>The product of (x) and (y), namely ([x_0 \cdot y_0, x_1 \cdot y_1, \ldots, x_{N/2-1} \cdot y_{N/2-1}])</td>
</tr>
<tr>
<td>(x + y)</td>
<td>The addition of (x) and (y), namely ([x_0 + y_0, x_1 + y_1, \ldots, x_{N/2-1} + y_{N/2-1}])</td>
</tr>
<tr>
<td>(x - y)</td>
<td>The subtraction of (x) and (y), namely ([x_0 - y_0, x_1 - y_1, \ldots, x_{N/2-1} - y_{N/2-1}])</td>
</tr>
</tbody>
</table>

### Algorithm 1: \(x = \text{Enc}(x)\).

```
Input: x
Output: x
(1) encode_double (x, Δ, x)
(2) encrypt (\(\langle x \rangle\), x)
(3) return: x
```

### Algorithm 2: \([x_0, x_1, \ldots, x_{n-1}] = \text{Dec}(X)\).

```
Input: X
Output: \([x_0, x_1, \ldots, x_{n-1}]\)
(1) for (\(i = 0\) to \(n - 1\)) do
(2) decrypt (\(X_i\), \(\langle x_i \rangle\))
(3) decode_double (\(\langle x_i \rangle\), \(x_i\))
(4) \(x_i = x_i\).get(0)
(5) end for
(6) return: \([x_0, x_1, \ldots, x_{n-1}]\)
```

### Algorithm 3: \(x \cdot y = \text{Mul}(x, y)\).

```
Input: x, y
Output: x \cdot y
(1) mod_switch_to_inplace (y, x.parms_id())
(2) multiply (x, y, x \cdot y)
(3) relinearize_inplace (x \cdot y, rk)
(4) rescale_to_next_inplace (x \cdot y)
(5) x \cdot y.set_scale (Δ)
(6) return: x \cdot y
```

### Algorithm 4: \(x \cdot y = \text{Mul Plain}(x, y)\).

```
Input: x, y
Output: x \cdot y
(1) encode_double (y, Δ, \(\langle y \rangle\))
(2) mod_switch_to_inplace (\(\langle y \rangle\), x.parms_id())
(3) multiply_plain (x, \(\langle y \rangle\), x \cdot y)
(4) rescale_to_next_inplace (x \cdot y)
(5) x \cdot y.set_scale (Δ)
(6) return: x \cdot y
```

### Algorithm 5: \(x + y = \text{Add}(x, y)\).

```
Input: x, y
Output: x + y
(1) mod_switch_to_inplace (y, x.parms_id())
(2) add (x, y, x + y)
(3) return: x + y
```

### Algorithm 6: \(x + y = \text{Add Plain}(x, y)\).

```
Input: x, y
Output: x + y
(1) encode_double (y, Δ, \(\langle y \rangle\))
(2) mod_switch_to_inplace (\(\langle y \rangle\), x.parms_id())
(3) add_plain (x, \(\langle y \rangle\), x + y)
(4) return: x + y
```
wants to outsource to SP to train a LR model without disclosing its own training data privacy. The specific description of the proposed P²OLR is as follows.

1. **DO generates** $sk, pk, rk, gk$, **computes** $l = 2m/N$, **calls** the Algorithm 1 to encrypt the training data $D$ into $l \times n$ ciphertexts.

**Algorithm 1:**

1. **Input:** $x, y$
2. **Output:** $y = \sigma(x)$
3. **1.** $x_{\text{inplace}}(y, x, \text{parms_id})$
4. **2.** $x - y$
5. **3.** $\text{return: } x$

**Algorithm 2:**

1. **Input:** $x, y$
2. **Output:** $x$
3. **1.** $x_{\text{inplace}}(y, x, \text{parms_id})$
4. **2.** $x - y$
5. **3.** $\text{return: } x$

**Algorithm 3:**

1. **Input:** $x = [x_0, x_1, \ldots, x_{N/2-1}]$
2. **Output:** $y = [\sum_{i=0}^{N/2-1} x_i, \sum_{i=0}^{N/2-1} x_i, \ldots, \sum_{i=0}^{N/2-1} x_i]$
3. **1.** $y = x$
4. **2.** for ($k = N/2; k \geq 1; k = k/2$) do
5. **3.** rotate_vector ($y, k, gk, z$)
6. **4.** add_inplace ($y, z$)
7. **5.** end for
8. **6.** $\text{return: } y$

**Figure 2:** Sigmoid approximation.

\[ y = \sigma(x) \]

\[ y = g(x) \]
\textbf{Algorithm 10: } $g(x) = \text{Sigmoid\_Approximation}(x)$.

\begin{align}
\llbracket y_0^T \rrbracket & = \text{Enc} \left( \llbracket y_0, y_1, \ldots, y_{N/2-1} \rrbracket \right), \\
\llbracket y_1^T \rrbracket & = \text{Enc} \left( \llbracket y_{N/2}, y_{N/2+1}, \ldots, y_{N-1} \rrbracket \right), \\
\llbracket y_{l-1}^T \rrbracket & = \text{Enc} \left( \llbracket y_{m-(l-1)/2}, y_{m-(l-1)/2+1}, \ldots, y_{m-1} \rrbracket \right), \\
\llbracket x_{0,1}^T \rrbracket & = \text{Enc} \left( \llbracket x_{0,1,1,1}, \ldots, y_{N/2-1,1} \rrbracket \right), \\
\llbracket x_{0,2}^T \rrbracket & = \text{Enc} \left( \llbracket x_{0,2,1,2}, \ldots, y_{N/2-1,2} \rrbracket \right), \\
\llbracket x_{1,1}^T \rrbracket & = \text{Enc} \left( \llbracket x_{N/2,1,2}, x_{N/2+1,1,2}, \ldots, y_{N-1,1} \rrbracket \right), \\
\llbracket x_{1,2}^T \rrbracket & = \text{Enc} \left( \llbracket x_{N/2,2,1,2}, x_{N/2+1,2,1,2}, \ldots, y_{N-1,2} \rrbracket \right), \\
\llbracket x_{2,1}^T \rrbracket & = \text{Enc} \left( \llbracket x_{N/2,2,2}, x_{N/2+1,2,2}, \ldots, y_{N-1,1} \rrbracket \right), \\
\llbracket x_{2,2}^T \rrbracket & = \text{Enc} \left( \llbracket x_{N/2,2,2}, x_{N/2+1,2,2}, \ldots, y_{N-1,2} \rrbracket \right).
\end{align}

(2)

\begin{align}
\llbracket a \rrbracket & = \text{Enc} \left( \llbracket a/m, 0, 0, \ldots, 0 \rrbracket \right). 
\end{align}

(4)

and sends $y_0^T, y_1^T, \ldots, y_{l-1}^T, x_{0,1}^T, x_{0,2}^T, x_{1,1}^T, \ldots, x_{2,2}^T, x_{0,0}^T, x_{0,1}^T, \ldots, x_{l-2,1}^T, x_{l-2,2}^T, x_{l-1,1}^T, \ldots, x_{l-1,2}^T, w_{0,0}^T, w_{0,1}^T, \ldots, w_{n-1}^T, a/m, N, Q, sk, pk, rk, gk, t$ to SP.

(2) SP computes ciphertexts

\begin{align}
\llbracket x_{i,0}^T \rrbracket & = \text{Enc} \left( \llbracket 1, 1, \ldots, 1 \rrbracket \right), \quad (i = 1, 2, \ldots, l - 2), \\
\llbracket x_{l-1,0}^T \rrbracket & = \text{Enc} \left( \llbracket 1, 1, \ldots, 1 \rrbracket \right), \\
\end{align}

(5) and sets the lists.
Security and Communication Networks

D takes the polynomial-modulus-degree maximum number of iterations

use the HEAAN library [33] to provide HE cryptographic

tions are publicly available at [35, 36], respectively, which

employ 5-fold cross-validation method to obtain the validity

proposed P2OLR with the related P2OLR [17, 18, 22]. We

implement all experiments on a 32-core Intel Xeon CPU

5. Performance Evaluation

We implement all experiments on a 32-core Intel Xeon CPU

with 256 GB RAM. We compare the performance of the proposed P^OLR with the related P^OLR [17, 18, 22]. We employ 5-fold cross-validation method to obtain the validity of the experimental results. For [17, 18], the implementations are publicly available at [35, 36], respectively, which use the HEAAN library [33] to provide HE cryptographic operations. For [22] and the proposed P^OLR, we employ the Microsoft SEAL library [26] for the HE cryptographic operations. For all experiments, we set the learning rate α = 0.01, random initial weight vector \{w^{(i)}_0, w^{(i)}_1, \ldots, w^{(i)}_{m-1}\} maximum number of iterations \( \lambda = 20 \), and scaling factor \( \Delta = 2^{40} \). To guarantee \( \kappa = 128 \) bit security, the scheme [17] takes the polynomial-modulus-degree \( N = 2^{17} \), coefficient-modulus \( Q \approx 2204 \) to 2406 bits; the scheme [18] sets the \( N = 2^{17} \), \( Q = 1176 \) bits; the scheme [22] chooses the \( N = 2^{17} \), \( Q = 320 \) bits; For the proposed P^OLR, we select \( N = 2^{15} \), \( Q = 512 \) bits. Using the three datasets [18]: D_1—Umaru Impact Study, D_2—Myocardial Infarction Study from Edinburgh, D_3—Nhanes III, we compare the proposed P^OLR with the related P^OLR [17, 18, 22] in terms of the encryption time (E. time) and decryption time (D. time) of DO, storage of encrypted training data, and training time (T. time). accuracy, precision, recall, F1-score and AUC of model. All comparison results are shown as an average of 10 experiments. The performance comparisons of the proposed P^OLR and the related P^OLR [17, 18, 22] are shown in Table 2.

From Table 2, we can see that, compared with the related P^OLR [17, 18, 22], the proposed P^OLR has a better performance. Specifically, as shown in Figure 3, under the training dataset D_1, the encryption time of DO in the proposed P^OLR is 2.01 s, which is reduced by nearly 71.4%, 7.8%, and 93.3% respectively compared with the encryption time of DO in [17, 18, 22]; under the training dataset D_2, the encryption time of DO in the proposed P^OLR is 2.16 s, which is reduced by nearly 73.6%, 2.3%, and 96.8% respectively compared with the encryption time of DO in [17, 18, 22]; under the training dataset D_3, the encryption time of DO in the proposed P^OLR is 3.49 s, which is reduced by nearly 75.9%, 81.6%, and 75.0% respectively compared with the encryption time of DO in [17, 18, 22].

As can be seen in Figure 4, under the training dataset D_1, the decryption time of DO in the proposed P^OLR is 0.23 s, which is reduced by almost 95.3% and 41.0% respectively in comparison to the decryption time of DO in [17, 18]; under the training dataset D_2, the decryption time of DO in the proposed P^OLR is 0.26 s, which is reduced by almost 95.0% and 36.6% respectively in comparison to the decryption time of DO in [17, 18]; under the training dataset D_3, the decryption time of DO in the proposed P^OLR is 0.45 s, which is reduced by almost 96.1% and 61.1% respectively in comparison to the decryption time of DO in [17, 18]. The decryption time of DO in [22] is smaller in comparison to that of the proposed P^OLR.

As described in Figure 5, under the training dataset D_1, the storage of encrypted training data in the proposed P^OLR is 72.00 MB, compared with the storage of encrypted training data in [17, 22], which is reduced by nearly 88.9% and 95.0% under the training dataset D_2, the storage of encrypted training data in the proposed P^OLR is 80.00 MB, compared with the storage of encrypted training data in [17, 22], which is reduced by nearly 89.0% and 97.4% under the training dataset D_3, the storage of encrypted training data in the proposed P^OLR is 128.00 MB, compared with the storage of encrypted training data in [17, 18, 22], which is reduced by nearly 89.4%, 13.0% and 99.7% respectively. Although the storage of encrypted training data for dataset D_1 and D_3 in [18] is smaller than that of the proposed P^OLR, as the number of samples m and features n increases, for dataset D_3, the storage of encrypted training data in the proposed P^OLR is smaller than that of [22].

As displayed in Figure 6, under the training dataset D_1, the training time of model in the proposed P^OLR is 2.64 min, which is reduced by almost 96.6%, 73.8%, and 90.1% respectively than the training time of model in [17, 18, 22]; under the training dataset D_2, the training time of model in the proposed P^OLR is 2.91 min, which is reduced by almost 96.5%, 71.7%, and 95.0% respectively than the training time of model in [17, 18, 22]; under the training dataset D_3, the training time of model in the proposed P^OLR is 4.21 min, which is reduced by almost 96.5%, 79.8%, and 99.4% respectively than the training time of model in [17, 18, 22].
In the proposed P2OLR, the average accuracy of model in [17, 18, 22] is 83.7%, which has nearly 4.6% improvement compared with the average accuracy of model in the proposed P2OLR is 90.6%, which has nearly 9.0%, 7.6%, and 7.9% improvement respectively compared with the average accuracy of model in [17, 18, 22].

As illustrated in Figure 7, under the training dataset $D_1$, the average accuracy of model in the proposed P2OLR is 80.6%, which has nearly 5.8%, 6.2%, and 6.2% improvement respectively compared with the average accuracy of model in [17, 18, 22]; under the training dataset $D_2$, the average accuracy of model in the proposed P2OLR is 90.6%, which has nearly 9.0%, 7.6%, and 7.9% improvement respectively compared with the average accuracy of model in [17, 18, 22]; under the training dataset $D_3$, the average accuracy of model in the proposed P2OLR is 83.7%, which has nearly 4.6%, 4.5%, and 5.8% improvement respectively compared with the average accuracy of model in [17, 18, 22].

As illustrated in Figure 8, under the training dataset $D_1$, the average precision of model in the proposed P2OLR is 95.6%, which has nearly 3.3%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22]; under the training dataset $D_2$, the average precision of model in the proposed P2OLR is 95.1%, which has nearly 5.4%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22];

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$m$</th>
<th>$n$</th>
<th>$\lambda$</th>
<th>Scheme</th>
<th>E. time (s)</th>
<th>D. time (s)</th>
<th>Storage (MB)</th>
<th>T. time (min)</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1-score (%)</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_1$</td>
<td>575</td>
<td>8</td>
<td>20</td>
<td>[17]</td>
<td>7.04</td>
<td>4.93</td>
<td>648.56</td>
<td>77.35</td>
<td>74.8</td>
<td>92.3</td>
<td>71.4</td>
<td>80.5</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[18]</td>
<td>2.18</td>
<td>0.39</td>
<td>36.75</td>
<td>10.09</td>
<td>74.4</td>
<td>90.9</td>
<td>71.4</td>
<td>80.0</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30.28</td>
<td>0.06</td>
<td>[22]</td>
<td>30.28</td>
<td>0.06</td>
<td>1438.75</td>
<td>26.67</td>
<td>74.4</td>
<td>90.9</td>
<td>71.4</td>
<td>80.0</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P2OLR</td>
<td>2.01</td>
<td>0.23</td>
<td>72.00</td>
<td>2.64</td>
<td>80.6</td>
<td>95.6</td>
<td>77.4</td>
<td>85.5</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.17</td>
<td>5.19</td>
<td>726.86</td>
<td>83.57</td>
<td>81.6</td>
<td>89.7</td>
<td>82.4</td>
<td>85.9</td>
<td>0.82</td>
</tr>
<tr>
<td>$D_2$</td>
<td>1253</td>
<td>9</td>
<td>20</td>
<td>[17]</td>
<td>8.71</td>
<td>5.19</td>
<td>726.86</td>
<td>83.57</td>
<td>81.6</td>
<td>89.7</td>
<td>82.4</td>
<td>85.9</td>
<td>0.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[18]</td>
<td>2.21</td>
<td>0.41</td>
<td>36.75</td>
<td>10.28</td>
<td>83.0</td>
<td>90.4</td>
<td>83.5</td>
<td>86.8</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>68.35</td>
<td>0.06</td>
<td>[22]</td>
<td>68.35</td>
<td>0.06</td>
<td>3133.75</td>
<td>57.35</td>
<td>82.7</td>
<td>90.4</td>
<td>82.9</td>
<td>86.5</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P2OLR</td>
<td>2.16</td>
<td>0.26</td>
<td>80.00</td>
<td>2.91</td>
<td>90.6</td>
<td>95.1</td>
<td>90.6</td>
<td>92.8</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14.48</td>
<td>11.65</td>
<td>1203.00</td>
<td>121.95</td>
<td>79.1</td>
<td>50.0</td>
<td>61.2</td>
<td>55.0</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13.96</td>
<td>0.48</td>
<td>147.00</td>
<td>20.86</td>
<td>79.2</td>
<td>50.2</td>
<td>61.3</td>
<td>55.2</td>
<td>0.71</td>
</tr>
<tr>
<td>$D_3$</td>
<td>15649</td>
<td>15</td>
<td>20</td>
<td>[17]</td>
<td>823.56</td>
<td>0.06</td>
<td>39123.75</td>
<td>718.25</td>
<td>77.9</td>
<td>52.4</td>
<td>62.2</td>
<td>56.9</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[18]</td>
<td>3.49</td>
<td>0.45</td>
<td>128.00</td>
<td>4.21</td>
<td>83.7</td>
<td>60.3</td>
<td>64.2</td>
<td>62.2</td>
<td>0.85</td>
</tr>
</tbody>
</table>

As illustrated in Figure 7, under the training dataset $D_1$, the average accuracy of model in the proposed P2OLR is 83.7%, which has nearly 4.6%, 4.5%, and 5.8% improvement respectively compared with the average accuracy of model in [17, 18, 22].

As illustrated in Figure 8, under the training dataset $D_1$, the average precision of model in the proposed P2OLR is 95.6%, which has nearly 3.3%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22]; under the training dataset $D_2$, the average precision of model in the proposed P2OLR is 95.1%, which has nearly 5.4%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22];

As illustrated in Figure 8, under the training dataset $D_1$, the average precision of model in the proposed P2OLR is 95.6%, which has nearly 3.3%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22]; under the training dataset $D_2$, the average precision of model in the proposed P2OLR is 95.1%, which has nearly 5.4%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22];

As illustrated in Figure 8, under the training dataset $D_1$, the average precision of model in the proposed P2OLR is 95.6%, which has nearly 3.3%, 4.7%, and 4.7% improvement respectively compared with the average precision of model in [17, 18, 22];
The encryption time of DO (s)
Dataset D1

(a)

The encryption time of DO (s)
Dataset D2

(b)

The encryption time of DO (s)
Dataset D3

(c)

Figure 3: The encryption time of DO.

The decryption time of DO (s)
Dataset D1

(a)

The decryption time of DO (s)
Dataset D2

(b)

Figure 4: Continued.
under the training dataset $D_3$, the average precision of model in the proposed P$^3$OLR is 60.3%, which has nearly 10.3%, 10.1%, and 7.9% improvement respectively compared with the average precision of model in [17, 18, 22].

As illustrated in Figure 9, under the training dataset $D_1$, the average recall of model in the proposed P$^3$OLR is 77.4%, which has nearly 6.0%, 6.0%, and 6.0% improvement respectively compared with the average recall of model in [17, 18, 22]; under the training dataset $D_2$, the average recall of model in the proposed P$^3$OLR is 90.6%, which has nearly 8.2%, 7.1%, and 7.7% improvement respectively compared with the average recall of model in [17, 18, 22]; under the training dataset $D_3$, the average recall of model in the proposed P$^3$OLR is 64.2%, which has nearly 3.0%, 2.9%, and 2.0% improvement respectively compared with the average recall of model in [17, 18, 22].

As illustrated in Figure 10, under the training dataset $D_1$, the average F1-score of model in the proposed P$^3$OLR is 85.5%, which has nearly 5.0%, 5.5%, and 5.5% improvement respectively compared with the average F1-score of model in [17, 18, 22]; under the training dataset $D_2$, the average F1-score of model in the proposed P$^3$OLR is 92.8%, which has nearly 6.9%, 4.0%, and 4.3% improvement respectively compared with the average F1-score of model in [17, 18, 22]; under the training dataset $D_3$, the average F1-score of model in the proposed P$^3$OLR is 62.2%, which has nearly 7.2%, 7.0%, and 5.3% improvement respectively compared with the average F1-score of model in [17, 18, 22].

As demonstrated in Figure 11, under the training dataset $D_1$, the AUC of model in the proposed P$^3$OLR is 0.73, compared with the AUC of model in [17, 18, 22], which has nearly 0.05, 0.08, and 0.07 improvement respectively; under the training dataset $D_2$, the AUC of model in the proposed P$^3$OLR is 0.88, compared with the AUC of model in [17, 18, 22], which has nearly 0.06, 0.02, and 0.02 improvement respectively; under the training dataset $D_3$, the AUC of model in the proposed P$^3$OLR is 0.85, compared with the AUC of model in [17, 18, 22], which has nearly 0.02, 0.14, and 0.14 improvement respectively.

6. Security Analysis

In a semi-honest adversary model, we assume that DO and SP hold the public key $pk$, relinearization key $rk$, galois key $gk$, and only DO holds the secret key $sk$. For our P$^3$OLR that evaluates deterministic function $f$, following the simulation-based paradigm [37], we consider the security model for security analysis, namely, DO encrypts its private data $x$ and sends $x$ to SP. SP performs the homomorphic operations on $x$ to obtain $y$, homomorphically evaluates $f(x)$ on $x$ to obtain $f(x)$, and sends $f(x)$ to DO. DO decrypts $f(x)$ and obtains $f(x)$.

**Theorem 1.** We assume that SP is a semi-honest entity and assume that DO and SP do not collude with each other. Let $x$ be a private data of DO. If the HE scheme [23] provides semantic security, after performing the homomorphic operations on $x$ and the evaluation of $f(x)$ on $x$, DO learns $f(x)$ but nothing else, SP learns nothing.

**Security Proof.** The security proof of the proposed P$^3$OLR follows the simulation-based paradigm [37]. Let the view of DO and SP during the evaluation be $\mathcal{V}_DO$ and $\mathcal{V}_SP$, respectively. The view $\mathcal{V}_SP$ of SP consists of $\{pk, rk, gk, x, y, f(x)\}$. We construct a simulator $\delta_{SP}$ as...
Figure 5: The storage of encrypted training data.
Figure 6: The training time of model.

Figure 7: The accuracy of model.
Figure 8: The precision of model.

Figure 9: Continued.
Figure 9: The recall of model.

Figure 10: The F1-score of model.
follows. \( S \) randomly chooses input data \( x', y', f(x') \). Then, \( S \) simulates \( V' \) by \( V'_S = \{ pk, rk, gk, x', y', f(x') \} \). Since the HE scheme [23] provides semantic security by assumption, \( V'_S \) and \( V' \) are indistinguishable. Therefore, the proposed P2OLR is secure against a semi-honest SP.

7. Conclusion

In this paper, we present a method for achieving a P2OLR on encrypted training data, which enables data owners to utilize the powerful storage and computing resources of cloud service providers for logistic regression analysis without exposing the privacy of training data. We take advantage of the batching technique and SIMD mechanism in HE to speed up the training process. On the three public datasets, compared with the related P2OLR schemes [17, 18, 22], the model training time of the proposed P2OLR is reduced by more than 71.7%, and the proposed P2OLR has over 4.5%, 3.3%, 2.0%, 4.0%, and 0.02 performance in terms of the accuracy, precision, recall, F1-score, and AUC of model. There are still some limitations in applying our scheme to arbitrary datasets and performing arbitrary number of iterations on encrypted training data. In the future, we will extend our scheme to efficiently support P2OLR with arbitrary number of iterations.

Appendix

(1) \texttt{key\_generation}(\texttt{params}) \rightarrow \{ \texttt{sk}, \texttt{pk}, \texttt{rk}, \texttt{gk} \}: Given the \texttt{poly\_modulus\_degree} \( N \) and \texttt{coeff\_modulus} \( Q \), it returns the secret key \( \texttt{sk} \), public key \( \texttt{pk} \), relinearization key \( \texttt{rk} \), galois key \( \texttt{gk} \).

(2) \texttt{encode\_double}(\texttt{x}, \Delta, \texttt{x}): Given the message vector \( \texttt{x} \in \mathbb{C}^{N/2} \) and scaling factor \( \Delta \), it expands \( \texttt{x} \) to \( \mathbb{R} \) by \( \pi^{-1}(\texttt{x}) \), scales \( \pi^{-1}(\texttt{x}) \) by \( \Delta \cdot \pi^{-1}(\texttt{x}) \), and outputs the plaintext \( \texttt{x} = \sigma^{-1}(\Delta \cdot \pi^{-1}(\texttt{x})) \in \mathbb{R} \).

(3) \texttt{decode\_double}(\texttt{x}, \texttt{x}): Given the plaintext \( \texttt{x} \), it computes
\[
\Delta \pi^{-1}(\texttt{x}) = \Delta \cdot \pi^{-1}(\texttt{x}) \in \mathbb{R},
\]
\[
\texttt{x} = \pi \cdot \pi^{-1}(\texttt{x}) \in \mathbb{C}^{N/2}.
\]
(4) encrypt (x, x): Given the plaintext x, it encrypts x into a ciphertext x, and outputs the ciphertext x.

(5) decrypt (x, x): Given a ciphertext x, it decrypts x into a plaintext x, and outputs the plaintext x.

(6) add (x, y, x + y): Given two ciphertexts x and y, it computes x + y and saves the result as a new ciphertext x + y.

(7) add_inplace(x, y): Given two ciphertexts x and y, it computes x + y and saves the result in ciphertext x.

(8) add_plain(x, y, x + y): Given a ciphertext x and a plaintext y, it computes x + y and saves the result as a new ciphertext x + y.

(9) sub(x, y, x − y): Given two ciphertexts x and y, it computes x − y and saves the result as a new ciphertext x − y.

(10) multiply(x, y, x * y): Given two ciphertexts x and y, it computes x * y and saves the result as a new ciphertext x * y.

(11) multiply_plain(x, y, x * y): Given a ciphertext x and a plaintext y, it computes x * y and saves the result as a new ciphertext x * y.

(12) mod_switch_to_inplace(x, y, parms_id()): Given a ciphertext/plaintext x, a levels y, and a levels parms_id() of ciphertext y, it switches the levels of x1 to y parms_id().

(13) relinearize_inplace(x, rk): Given a ciphertext x and a relinearization_key rk, it relinearizes x and saves the result in ciphertext x.

(14) rescale_to_next_inplace(x): Given a ciphertext x, it switches the modulo of x to the next levels, reduces the length of the plaintext accordingly, and saves the result in ciphertext x.

(15) set_scale(Δ): Given a scaling factor Δ, it scales the ciphertext x by computing x.set_scale(Δ), and outputs the ciphertext x.

(16) rotate_vector(x, k, gk, y): Given a ciphertext x = [x0, x1, ..., xN/2−1], a rotation value k, and galois_key gk, it rotates x left by k, and saves the result as a new ciphertext y = [xk, xk+1, ..., xN/2−1, x0, x1, ..., xk−1].

Data Availability

Previously reported Umaru Impact Study, Myocardial Infarction dataset from Edinburgh and Nhanes III datasets were used to support this study and are available at https://doi.org/10.1186/s12920-018-0401-7. These prior studies (and datasets) are cited at relevant places within the text as references [18].

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant no. U19B2021.

References


