Privacy-preserving Logistic Regression Model Training Scheme by Homomorphic Encryption

Weijie Miao[1,2] and Wenyuan Wu*[1,2]

[1] Chongqing Key Laboratory of Secure Computing for Biology, Chongqing Institute of Green and Intelligent Technology,China
[2] Chongqing School, University of Chinese Academy of Sciences,China
{miaoweijie,wuwenyuan}@cigit.ac.cn

Abstract. In the field of big data, logistic regression for binary and multi-class classification is widely used. Nowadays, there is growing concern about data privacy protection issues. This paper focuses on scenarios involving two parties participating and data being horizontally distributed. Based on homomorphic encryption, a logistic regression model training scheme is designed. This scheme reduces the number of iterations in the training process by using the second-order approximation Newton's method. It employs the conjugate gradient method to solve the Newton's method, and introduces a small amount of interaction to reduce ciphertext domain division operations, greatly reducing the computational overhead of the ciphertext domain. Additionally, a new encoding method is used to reduce the number of ciphertext multiplications and communication overhead. Furthermore, the "One-vs-Rest" decomposition strategy is adopted, combined with SIMD(Single Instruction, Multiple Data) technology, extending the binary model to multi-classification. Experimental results show that with the use of Privacy-preserving Logistic Regression scheme (PPLR), for most datasets, setting the number of iterations to within 3 rounds can achieve accuracy comparable to existing privacy protection schemes of 5 to 7 rounds. For sample datasets with 60 and 112 dimensions, existing similar schemes require 90s and 165s to complete 5 rounds of iteration, while this scheme only requires 8s and 27s. Moreover, the communication overhead is reduced to half of the original scheme, requiring only 30.8MB and 62.7MB to complete training.

Keywords: Privacy protection· Newton-conjugate gradient method · Lgistic regression · Homomorphic encryption· CKKS.

1 Introduction

Currently, protecting data privacy is often achieved through the design of secure multi-party computation(MPC) and privacy protection protocols, and homomorphic encryption , differential privacy,oblivious transfer, garbled circuits, secret sharing et al. are commonly used privacy-enhancing technologies. Secure multi-party computation is often constrained by significant communication overhead, and some schemes rely on the assumption of trusted third parties, which

poses risks of privacy leakage[1].Differential privacy achieves privacy protection by adding noise to the dataset, rendering specific data meaningless while preserving the utility of statistical information. Since any computation performed on the altered data is only statistically correct, as noise increases, the predictive accuracy of the model decreases, and consequently, the utility of the model diminishes[2].Indeed, various other privacy protection technologies, such as oblivious transfer, garbled circuits, secret sharing, also face challenges such as high computational overhead, large communication costs, and limitations in application scenarios[3].Homomorphic encryption(HE), which supports computations directly on ciphertexts, yields results that are consistent with those obtained by performing the same operations on plaintexts. By operating on encrypted data without the need for decryption, HE guarantees data privacy while achieving accurate training of machine learning models[4].Among common privacy-enhancing technologies, it boasts higher security due to this property.This overcomes the problem of data non-sharability, providing an innovative solution for data mining involving sensitive information held by multiple parties.

Logistic regression is widely used in various fields such as data mining, automatic disease diagnosis, and economic forecasting.It is characterized by relatively low computational costs, and its principles and derivation process are easy to understand and implement. Logistic regression is commonly used for binary classification tasks; however, it has wide applications in multi-class tasks as well.In order to ensure the security and accuracy of the model training mentioned above, HE technology can be applied to logistic regression. However, due to the significant computational overhead associated with bootstrapping technology, the more efficient Leveled HE can only calculate functions of limited depth, and most current homomorphic encryption technologies do not support efficient ciphertext division operations, the application of HE in model training is relatively limited.

This paper addresses the common issues of high iteration counts, large computational overhead, and high communication costs in existing research. Specifically targeting scenarios where datasets are horizontally distributed across two users, we propose and implement a homomorphic encryption-based Newton-Conjugate Gradient (NCG) logistic regression solution. This approach reduces the number of iterations, lowers communication overhead, and enhances computational efficiency by minimizing homomorphic ciphertext division through a small amount of interaction. The specific contributions are as follows:

- By utilizing second-order approximation Newton's method,a secure logistic regression approach is proposed, which reduces the number of iterations during training. The conjugate gradient method is used to solve the Newton method update direction through a small number of iterations to achieve dynamic update of the Hessian matrix.
- Solve the problem of low efficiency of ciphertext division operation involved in the inversion process of Hessian matrix through a small amount of interaction,and use a new encoding method to reduce the number of ciphertext multiplications and communication overhead.

– Adopt OvR(One-vs-Rest) decomposition strategy combined with SIMD (Single Instruction, Multiple Data) technology to train multiple classifier. One ciphertext packages data of multiple binary classification models, trains multiple classifiers in parallel,thereby extending the binary classification model to multi-class classification.

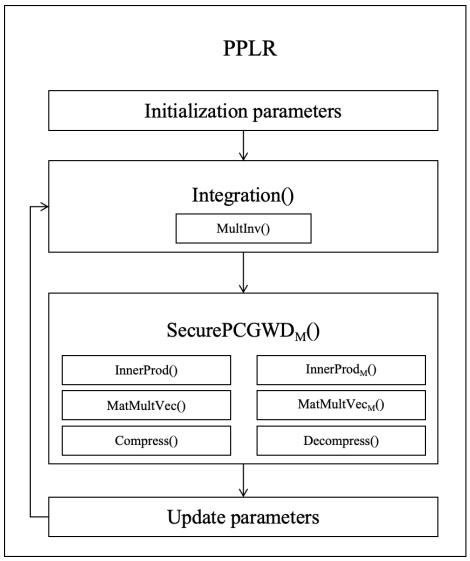

Fig. 1. Algorithm structure.

2 Related Work

In 1978, the concept of Homomorphic Encryption (HE) was first proposed[5]. Subsequently, an increasing number of researchers have been involved in the design of homomorphic encryption schemes, and various research proposals based on homomorphic encryption for secure logistic regression models have been put forward.

Shi[6]proposed a secure multi-party computation framework SMAC-GLORE suitable for grid logistic regression. However, due to the limitations of garbled circuits, SMAC-GLORE faces challenges when dealing with a large number of records and features.Xie[7] proposed PrivLogit, which utilizes Yao's garbled circuits and Paillier encryption techniques. However, the computational cost of intermediate results in this scheme is quite expensive.References [8]-[12] all utilize homomorphic encryption schemes for training logarithms, but they all involve using polynomial approximation methods to evaluate non-linear functions in ma-

chine learning algorithms. Additionally, they all conduct centralized computations on a single dataset, without involving scenarios with multiple participants.

In[13][14] , logistic regression schemes for two-party computation in the federated learning scenario were proposed. However, these schemes all rely on a trusted third party, posing potential risks of privacy leakage.References [15]-[18]abandoned the assumption of a trusted third party, but its applicable scenarios mainly involved vertically distributed data.

Mohassel et al.[19]proposed a protocol, which utilizes mini-batch gradient descent. However, it suffers from slow convergence speed, requiring more iterations.Ghavamipour et al.[20]addressed the issue of limited convergence speed in logistic regression by employing the Newton-Raphson method. Their approach utilizes secret sharing and multi-party computation techniques to compute the inverse of matrices. However, in protocols based on multi-party secure computation, multiplication operations require multiple rounds of communication, thereby increasing communication overhead. Additionally, they adopted a fixed Hessian matrix approach, which may result in some loss of precision.

In fact, most of the aforementioned literature discusses scenarios related to binary classification, with limited research on multiclass logistic regression. Reference [21]analyzes multiclass models and designs a kNN (k-nearest neighbors) privacy-preserving solution for multiclass problems. References [22][23] focuses on designing a solution for privacy-preserving multiclass logistic regression. It employs the OvR strategy to extend binary classification to multiclass. However, both approaches use polynomial approximation methods and gradient descent, resulting in slow convergence speeds.

3 Preliminaries

3.1 Newton's Method for Solving Logistic Regression

Logistic regression is considered a convex optimization problem [24], widely used as a machine learning model for predicting the probability of events. Common methods for solving logistic regression include gradient descent, Newton's method et al. Newton's method is an approximation method used to find the roots of an equation in both real and complex domains. The method utilizes the first few terms of the Taylor series of the function $f(x)$ to search for the root of the equation $f(x) = 0$,the iteration formula is as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1}$$

In solving logistic regression parameters, the negative log-likelihood function of logistic regression is commonly employed as the optimization objective function. Due to the fact that the derivative value $f'(x)$ equals zero at points where the function takes extremal values, Newton's method can be used to find the zeros of $f'(x)$,which means Newton's method updates parameters iteratively using both the first and second derivatives of the function [25]. It converges faster

compared to gradient descent. The iteration formula is as follows:

$$\theta_{new} = \theta_{old} - \frac{f'(\theta_{old})}{f''(\theta_{new})} \tag{2}$$

Among them, $f'(\theta_{old})$ is the first derivative of the likelihood function, and $f''(\theta_{new})$ is the second derivative of the likelihood function. For logistic regression, the first derivative (gradient) and second derivative (Hessian matrix) of the likelihood function are:

$$\nabla J(\theta) = \sum_{i=1}^{m} (h_\theta(x_i) - y_i)x_i \tag{3}$$

$$H(\theta) = \sum_{i=1}^{m} h_\theta(x_i)(1 - h_\theta(x_i))x_i^T x_i \tag{4}$$

Among them, m is the number of samples and $h_\theta(x_i)$ is the hypothesis function of logistic regression, that is:

$$\theta_{new} = \theta_{old} - H(\theta_{old})^{-1}\nabla J(\theta_{old}) \tag{5}$$

After deformation, formula (4) can be obtained:

$$H(\theta_{old})(\theta_{old} - \theta_{new}) = \nabla J(\theta_{old}) \tag{6}$$

The transformed equation satisfies the form $A\beta = b$, where A represents the Hessian matrix$H(\theta_{old})$,b represents gradient$\nabla J(\theta_{old})$, β represents $(\theta_{old} - \theta_{new})$. Since logistic regression is a convex optimization problem, $H(\theta_{old})$ is symmetric positive-definite matrix, so θ_{new} can be solved using the conjugate gradient method [26], thus avoiding the problem of inverting the Hessian matrix during the Newton method solving process.

3.2 Multiclass Logistic Regression

Logistic regression can also be used to perform multiclass tasks. One of the common methods is based on the decomposition approach called "One-vs-Rest" (OvR) strategy. It utilizes binary logistic regression models to address multiclass problems[22].

For a given dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$,
$y_i \in C_1, C_2, \ldots, C_k$, the OvR strategy involves training k classifiers, where each classifier is trained to distinguish one class as positive and the rest of the classes as negative. After training, for a given input x_i, each classifier predicts the probability of $x_i$belonging to its corresponding class$\{P_1, P_2, \ldots, P_k\}$. Then, the confidence scores from each classifier are compared, and the class with the highest confidence score is selected as the predicted class label.This method only requires training k classifiers, which typically results in smaller storage and testing time overhead compared to other decomposition methods.

3.3 Conjugate Gradient Method

Conjugate Gradient Method is an iterative method used for solving linear systems of equations and optimization problems, commonly employed to solve linear systems of the form $A\beta = b$.In practical applications, some datasets may utilize pre-conditioned conjugate gradient method, which accelerates the convergence of the algorithm by selecting a pre-conditioning matrix to transform the system of equations into a well-conditioned equivalent system.The original conjugate gradient method requires two divisions in each iteration after the first iteration. Reference [27] proposed a modification of the original conjugate gradient method called the PCGWD (Pre-conditioned Conjugate Gradient with Division Delay) method, which delays the first division operation, allowing both division operations in each iteration to be performed simultaneously.

3.4 CKKS Homomorphic Encryption Scheme

Homomorphic encryption allows computations to be performed on encrypted data, resulting in the same computation outcomes as if the computations were performed on the plaintext data.The CKKS encryption scheme proposed by Cheon et al. [28] is a type of leveled homomorphic encryption technique that supports approximate arithmetic operations. Solving linear systems of equations of the form $A\beta = b$ involves floating-point arithmetic, making the CKKS encryption scheme suitable for such applications.The CKKS scheme is based on the Ring-LWE (Learning With Errors) problem, where encryption noise is incorporated as part of the error in the approximate computation process.

4 Privacy-Preserving Logistic Regression

4.1 Data Encoding and Encryption Methods

In [27], a method is employed where the matrix is packed into n ciphertexts and represented by a single ciphertext vector to encode the encrypted matrix. For a linear system $A\beta = b$ where the original matrix is $A_{n\times n} = (a_1, a_2, \cdots, a_n)$, it is encoded by column vectors. For a column vector a_i, firstly, its dimension is expanded to $2^{\lceil \log n \rfloor}$ by padding with zeros, denoted as $a_i^T = (A_{1i}, A_{2i}, \cdots, A_{ni}, 0, \cdots, 0)$. Then it is encrypted and packed into n ciphertexts, defined as :

$$ct_a \longleftarrow [Enc(a_1), Enc(a_2), \cdots, Enc(a_n)] \tag{7}$$

where $ct_{a[i]} \longleftarrow Enc(a_i)_{1\leq i\leq n}$.

Unlike the encoding method used in [27], this paper employs a more efficient encoding approach. The matrix is divided into multiple fixed-sized matrices, ensuring that each matrix is packed as a single ciphertext in row-wise fashion. If the data size cannot be evenly divided by the pre-set fixed-sized matrices, zeros can be used for padding.For the linear system $A\beta = b$, the original matrix $A_{n\times n}$

is packed into a single ciphertext. The structure of the corresponding ciphertext ct_a is as

$$ct_a \longleftarrow Enc(a_{00}, \cdots, a_{0n}, a_{10}, \cdots, a_{1n}, \cdots, a_{n0}, \cdots, a_{nn}) \tag{8}$$

The encoding method for vectors b and the target vector β satisfies:

$$ct_b \longleftarrow Enc(b_0, b_1, \cdots, b_n, \cdots, b_0, b_0, \cdots, b_n) \tag{9}$$

$$ct_\beta \longleftarrow Enc(\beta_0, \beta_1, \cdots, \beta_n, \cdots, \beta_0, \beta_1, \cdots, \beta_n) \tag{10}$$

The aforementioned encoding method allows for packing multiple plaintext data into a single ciphertext, leveraging SIMD (Single Instruction, Multiple Data) technology to significantly enhance training speed,which means matrix-vector multiplication requires only one multiplication operation to complete.

4.2 Homomorphic Operations in the Ciphertext Domain

To implement the conjugate gradient algorithm in the ciphertext domain, it is necessary to define operations for vector inner product and matrix-vector multiplication in the ciphertext domain. Although the CKKS homomorphic encryption scheme cannot directly compute vector inner products in the ciphertext domain, it can be achieved through rotation, addition, and multiplication operations(Algorithm InnerProd) [27].Conjugate gradient method requires only one matrix-vector multiplication per iteration. Based on the data encoding and encryption method described in [27], and utilizing the fundamental ciphertext computation provided by CKKS, the matrix-vector multiplication operation can be designed(Algorithm MatMultVec)[27] .

Based on the data encoding and encryption method proposed in this paper, utilizing the basic ciphertext computation provided by CKKS, we design computations for vector inner product and matrix-vector multiplication as shown in Algorithm 1 and Algorithm 2 . The encoding method for vectors ensures the following properties:$ct_{b'} = Enc(b_0, b_0, \cdots, b_0, b_1, b_1, \cdots, b_1, \cdots, b_n, b_n, \cdots, b_n)$,$ct_{\beta'} = Enc(\beta_0, \beta_0, \cdots, \beta_0, \beta_1, \beta_1, \cdots, \beta_1, \cdots, \beta_n, \beta_n, \cdots, \beta_n)$.

Furthermore, during the operation of the conjugate gradient method, division operations are involved. However, CKKS homomorphic encryption technology only provides limited homomorphic multiplicative inverse operations.

4.3 Ciphertext Domain Conjugate Gradient Method

This paper adopts the interactive secure multiplicative inverse protocol(Algorithm 3) proposed in [27] to achieve efficient division operations in the ciphertext domain, thereby assisting in completing the training process of the model.

To reduce communication overhead, this paper employs the ciphertext compression algorithm (Algorithm 4) designed in [27]. This algorithm compresses multiple ciphertexts that originally required two interactions into a single ciphertext for transmission. Leveraging the encoding characteristics of CKKS, the

Algorithm 1: InnerProd$_M(ct_{\beta'}, ct_{b'})$

input : Cipher text $ct_{\beta'}, ct_{b'}$ of vector β', b'
output: Inner product$c_{dot'} \longleftarrow Enc(\langle \beta', b' \rangle)$

$ct_{dot'} \longleftarrow Mult(ct_{\beta'}, ct_{b'})$;
for $i \leftarrow 0$ **to** $\lceil \log n \rfloor - 1$ **do**
  $ct_{dot'} \longleftarrow Add(ct_{dot'}, Rotate(ct_{dot'}; 2^j \times n))$;
end
return $ct_{dot'} \longleftarrow Enc(\sum_{i=0}^{i \leq n} b_i\beta_i, \sum_{i=0}^{i \leq n} b_i\beta_i, \cdots, \sum_{i=0}^{i \leq n} b_i\beta_i)$

Algorithm 2: MatMultVec$_M(ct_a, ct_x)$

input : Ciphertext ct_a, ct_x of matrix $A_{n\times n}$ and vector $x_{n\times 1}$
output: Ciphertext matrix-vector product ct_{Ax}

Select mask matrix $M_{n\times n} = \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{pmatrix} = \begin{pmatrix} 1\ 0 \cdots 0 \\ 1\ 0 \cdots 0 \\ \vdots\ \vdots\ \ddots\ 0 \\ 1\ 0 \cdots 0 \end{pmatrix}$;
$ct_{Ax} \longleftarrow Enc(0, \cdots, 0), c \longleftarrow Encode(M; p)$;
$ct_{tmp} \longleftarrow InnerProd(ct_a, ct_x)$;
$ct_{Ax} \longleftarrow CMult(ct_{tmp}; c)$;
for $i \leftarrow 1$ **to** $\lceil \log n \rfloor - 1$ **do**
  $ct_{Ax} \longleftarrow Add(ct_{Ax}, Rotate(ct_{Ax}; 2^j))$
end
return ct_{Ax}

Algorithm 3: MultInv(ct_{tmp})

input : $ct \longleftarrow Enc(x_1, x_2, \cdots, x_n)$
output: $ct \longleftarrow Enc(\frac{1}{x_1}, \frac{1}{x_2} \cdots, \frac{1}{x_n})$

Alice: uniformly and randomly select a real number vector $t = (t_1, t_2, \cdots, t_n)$ from a discrete distribution $\{1, 1.0001, \cdots, 2^4\}$ with an interval of 10^{-4} between adjacent points, let $ct \longleftarrow CMult(t \cdot 2^{\log p}, ct)$, and send ct to Bob;
Bob:$(t_1x_1, t_2x_2, \cdots, t_nx_n) \longleftarrow Dec(ct)$,$ct \longleftarrow Enc(\frac{1}{t_1x_1}, \frac{1}{t_2x_2} \cdots, \frac{1}{t_nx_n})$,and send ct to Alice;
Alice:$ct \longleftarrow CMult(t \cdot 2^{\log p}, ct)$;
return ct

Algorithm 4: Compress(ct_a, ct_b)

input : Ciphertext $ct_a \longleftarrow Enc(a, a, \cdots, a), ct_b \longleftarrow Enc(b, b, \cdots, b)$ of real numbers a, b
output: Compressed ciphertext $ct_{res} \longleftarrow Enc(a, 0, b, 0, \cdots, 0)$

Let $m_1 = (1, 0, \cdots, 0)$,$m_2 = (0, 0, 1, 0, \cdots, 0)$;
$ct_{res} \longleftarrow Add(CMult(ct_a, m_1), CMult(ct_b, m_2))$;
return ct_{res}

Algorithm 5: Decompress(ct_{res})

input : Compressed ciphertext $ct_{res} \longleftarrow Enc(a, 0, b, 0, \cdots, 0)$ of real numbers a, b

output: Decompressed ciphertext $ct_a \longleftarrow Enc(a, a, \cdots, a), ct_b \longleftarrow Enc(b, b, \cdots, b)$

Let $m_1 = (1, 0, \cdots, 0)$,$m_2 = (0, 0, 1, 0, \cdots, 0)$;
$ct_a \longleftarrow CMult(ct_{res}, m_1), ct_b \longleftarrow CMult(ct_{res}, m_2)$;
for $i \leftarrow 1$ **to** $\lceil \log n \rfloor - 1$ **do**
  $ct_a \longleftarrow Add(ct_a, Rotate(ct_a; 2^j))$
end
return ct_a, ct_b

effective information of the ith ciphertext is compressed into the 2^{i-1} plaintext slot of the output ciphertext, facilitating ciphertext decompression(Algorithm 5).

The process of solving the logistic regression model using the Newton method involves the issue of inverting the Hessian matrix. However, CKKS homomorphic encryption technology does not support efficient matrix inversion operations. Additionally, the conjugate gradient method is an iterative approach that differs from operations like Gaussian elimination or QR decomposition, which require multiple rounds of computation. The conjugate gradient method can achieve a certain level of accuracy with fewer iterations. Therefore, in this paper, we adopt the conjugate gradient method to solve for the Newton update direction. This paper proposes a ciphertext domain conjugate gradient algorithm(Algorithm 6) with division delay, which is based on an improved conjugate gradient method incorporating division operations.

The algorithm packs the matrix into a single data stream, thereby significantly reducing the number of multiplication operations. For $* \in \{a, b, \beta, M^{-1}\}$,the ciphertext form must satisfy:

$$ct_{*'} \longleftarrow Enc(*_0, \cdots, *_0, *_1, \cdots, *_1, \cdots, *_n, \cdots, *_n) \tag{11}$$

$$ct_{*} \longleftarrow Enc(*_0, *_1, \cdots, *_n, \cdots, *_0, *_1, \cdots, *_n,) \tag{12}$$

4.4 Newton-Conjugate Gradient Logistic Regression Scheme in Ciphertext Domain

In the scenario involving two parties, Alice provides the homomorphic encryption keys and is responsible for the ciphertext refresh operation, while Bob is responsible for invoking the core computation module. Both parties aggregate data to obtainct_a, ct_b, ct_M (Algorithm 7) which are then passed into the ciphertext domain conjugate gradient algorithm to solve for the Newton update direction. This process iterates continuously until the stopping condition is met, completing the model training. The ciphertext domain Newton-Conjugate Gradient logistic regression algorithm is illustrated in Algorithm 8.

Algorithm 6: SecurePCGWD$_M(ct_{b'}, ct_b, ct_{M^{-1'}}, ct_{M^{-1}}, ct_{a'})$

input : Ciphertext $ct_{b'}, ct_b, ct_{M^{-1'}}, ct_{M^{-1}}, ct_{a'}$
output: Updated parametersβ

Bob initializes $k = 0, \varepsilon = 10^{-4}, k_{max} = n, ct_r \longleftarrow ct_b, ct_{r'} \longleftarrow ct_{b'}$;
while $k \leq k_{max}$ **do**
  $k = k + 1, ct_z \longleftarrow Mult(ct_r, ct_{M^{-1}}), ct_{z'} \longleftarrow Mult(ct_{r'}, ct_{M^{-1'}})$;
  if $k == 1$ **then**
    $ct_p \longleftarrow ct_z, ct_{p'} \longleftarrow ct_{z'}$;
    $ct_{\mu_1} \longleftarrow InnerProd(ct_z, ct_r)$;
    $ct_\omega \longleftarrow MatMultVec_M(ct_{a'}, ct_p)$;
    $ct_{p\omega} \longleftarrow InnerProd_M(ct_\omega, ct_p)$;
    $ct_\alpha \longleftarrow Mult(ct_{\mu_1}, MultInv(ct_\alpha, ct_{p\omega}))$;
    $ct_\beta \longleftarrow Add(ct_\beta, Mult(ct_\alpha, ct_p))$;
    $ct_r \longleftarrow Sub(ct_{r'}, Mult(ct_\alpha, ct_\omega))$;
  else
    $ct_{\mu_2} \longleftarrow ct_{\mu_1}$;
    $ct_{\mu_1} \longleftarrow InnerProd(ct_z, ct_r), ct_f \longleftarrow Mult(Mult(ct_{\mu_2}, ct_z), Mult(ct_{\mu_1}, ct_p))$;
    $ct_{f'} \longleftarrow Mult(Mult(ct_{\mu_2}, ct_{z'}), Mult(ct_{\mu_1},$ ct$_{p'}$)), $ct_\omega \longleftarrow MatMultVec_M(ct_{a'}, ct_f)$;
    $ct_{\omega f} \longleftarrow InnerProd_M(ct_\omega, ct_{f'})$;
    $ct_{tmp} \longleftarrow Compress(ct_{f\omega}, ct_{\mu_2})$;
    $ct_{tmp^{-1}} \longleftarrow MultInv(ct_{tmp})$;
    $ct_{f\omega^{-1}}, ct_{\mu_2^{-1}} \longleftarrow Decompress(ct_{tmp})$;
    $ct_\alpha \longleftarrow Mult(ct_{\mu_1}, ct_{f\omega^{-1}})$;
    $ct_p \longleftarrow Mult(ct_f, ct_{\mu_2^{-1}})$;
    $ct_\beta \longleftarrow Add(ct_\beta, Mult(ct_{\mu_2}, Mult(ct_\alpha,$ ct$_f$))), $ct_r \longleftarrow Sub(ct_{r'}, Mult(ct_{\mu_2}, Mult($ ct$_\alpha, ct_\omega$)));
  end
  $ct_{r^2} \longleftarrow Mult(ct_r, ct_r)$;
  for $i \leftarrow 1$ **to** $\lceil \log n \rfloor - 1$ **do**
    $ct_{r^2} \longleftarrow Add(ct_{r^2}, Rotate(ct_{r^2}; 2^j \times n))$
  end
  Randomly select mask vector u;
  $ct_r \longleftarrow Add(u, ct_r), ct_p \longleftarrow Add(u, ct_p)$;
  Alice send ct_r, ct_{r^2}, ct_p to Bob;
  Bob refreshct_r, ct_p compute $\|r\|_2$;
  Bob send $ct_r, ct_p, ct_{r'}, ct_{p'}, \|r\|_2$to Alice;
  $ct_r \longleftarrow Sub(ct_r, u), ct_p \longleftarrow Sub(ct_p, u)$;
end
Alice send $ct_\beta$to Bob;
Bob calculate $\beta \longleftarrow Dec(ct_\beta)$,send β to Alice;
return β

If the algorithm is called without providing parameters $ct_{M^{-1'}}, ct_{M^{-1}}$, it defaults to using both $ct_{M^{-1'}}$ and $ct_{M^{-1}}$ as matrices filled with ones.This corresponds to the original conjugate gradient method. If $ct_{M^{-1'}}, ct_{M^{-1}}$ are provided, it represents preconditioned conjugate gradient method.The preconditioned matrix in this paper is the inverse of the diagonal elements of matrix A.

Algorithm 7: Integration$(X_1, Y_1, X_2, Y_2, \beta)$

input : $X_1, Y_1, X_2, Y_2, \beta$
output: $ct_a, ct_{a'}, ct_b, ct_{b'}, ct_M, ct_{M^{-1}}, ct_{M^{-1'}}$

Alice: Initializes the encryption system, generates public and private keys, and sends the public key to Bob;
Alice: $A_1 = \sum_{i=1}^{m} h_\beta(x_{1i})(1-h_\beta(x_{1i}))x_{1i}^T x_{1i}$, $b_1 = \sum_{i=1}^{m}(h_\beta(x_{1i}) - y_{1i})x_{1i}$, extracts the diagonal elements of A_1 to form a vector M_1;
Alice:Encrypts matrices ct_{A_1}, ct_{b_1},$ct_{A_1'}$,$ct_{b_1'}$, and vector ct_{M_1}, and sends them to Bob;
Bob: $A_2 = \sum_{i=1}^{m} h_\beta(x_{2i})(1-h_\beta(x_{2i}))x_{2i}^T x_{2i}$, $b_2 = \sum_{i=1}^{m}(h_\beta(x_{2i}) - y_{2i})x_{2i}$,then extracts the diagonal elements of A_2 to form a vector M_2;
Bob:$ct_a \longleftarrow Add(ct_{A_1}, A_2), ct_b \longleftarrow Add(ct_{b_1}, b_2), ct_{a'} \longleftarrow Add(ct_{A_1'}, A_{2'}), ct_{b'} \longleftarrow Add(ct_{b_1'}, b_{2'}), ct_M \longleftarrow Add(ct_{M_1}, M_2)$;
Bob: $ct_{M^{-1}}$,$ct_{M^{-1'}}$,$\longleftarrow MultInv(ct_M)$;
return $ct_a, ct_{a'}, ct_b, ct_{b'}, ct_M, ct_{M^{-1}}, ct_{M^{-1'}}$

Algorithm 8: PPLR

input : X_1, Y_1, X_2, Y_2
output: Model update parameters β

Initializes $\beta \longleftarrow (0, 0, \cdots, 0)$;
while $q \leq q_{max}$ **do**
  $ct_a, ct_{a'}, ct_b, ct_{b'}, ct_M, ct_{M^{-1}}, ct_{M^{-1'}} \longleftarrow Integration(X_1, Y_1, X_2, Y_2, \beta), q = q + 1,$;
  Bob: $ct_\beta \longleftarrow SecurePCGWD_M(ct_{b'}, ct_b, ct_{M^{-1'}}, ct_{M^{-1}}, ct_{a'})$
  Bob: Sends ct_β to Alice;
  Alice: $\beta \longleftarrow Dec(ct_b)$, sends to Bob;
  Both parties update parameters β;
end
return β

For multi-class logistic regression tasks using the OvR strategy, one only needs to invoke Algorithm PPLR to train multiple binary logistic regression classifiers. Then, based on the confidence scores from these classifiers, the final classification category can be determined. This approach extends the binary

classification model to a multi-class classification model.In addition, since the dimensionality n of the dataset is much smaller than the number of ciphertext slots $N/2$ in the CKKS scheme, Alice can pack the data used for training multiple classifiers into a single ciphertext. By employing SIMD technique, can effectively utilize the ciphertext slots, completing the training of multiple classifiers in one go. This approach significantly reduces time overhead, as it allows for parallel processing of multiple classifiers simultaneously.

5 Analysis of the Scheme

5.1 Security Analysis

In the proposed scheme, the two users participating in model training, Alice and Bob, behave as semi-honest entities. Their common goal is to obtain a correct set of model parameters, thus excluding the presence of malicious entities. To ensure the privacy of users' original data, data encryption is employed, and the majority of computational operations are performed on ciphertexts. Homomorphic encryption technology is utilized to safeguard the privacy of user data, with the security of the CKKS homomorphic encryption scheme resting on the computational hardness of the Ring Learning with Errors(R-LWE) problem. Detailed security analysis can be found in [28][31].During the execution of the ciphertext domain logistic regression algorithm, the transmission of ciphertexts r, p, $f\omega$ and μ_2 is necessary. Particularly, during the computation of the multiplicative inverse operation, Alice and Bob need to engage in one interaction involving $f\omega$ and μ_2. To mitigate potential risks of information leakage, specific domain-specific multiplication perturbations are introduced to the original data in ciphertext state, ensuring that the protocol possesses sufficient security.Assuming that the elements h of the matrices and vectors involved fall within the range of $[-2^{16}, 2^{16}]$, satisfying general matrix operation requirements, and uniformly selecting random perturbations t from a discrete distribution $\{1, 1.0001. \cdots, 2^4\}$ with intervals of 10^{-4} between adjacent points, we can form a plaintext message space $h \cdot t \in [-2^{20}, 2^{20}]$.

In a secure multiplicative inverse protocol, Bob's probability of successfully guessing the random perturbation t is only $\frac{1}{150000}$. Additionally, since the dimensionality of the dataset n is much smaller than the number of ciphertext slots $N/2$ in the CKKS scheme, Alice can choose the encryption positions of the data, while the remaining ciphertext slots are filled with random numbers. This means that even if Bob successfully guesses the random perturbation, he still needs to guess the positions of the data.Therefore, trying to infer the other party's data is very difficult in this scheme.

To address the issue of ciphertext noise expansion during iterations, noise reduction processing is performed on two intermediate ciphertexts after each iteration. The specific implementation involves both parties engaging in an interaction where the owner of the encryption keys decrypts these two ciphertexts and then re-encrypts them.This strategy effectively reduces the noise in the ci-

phertexts, minimizing the ciphertext modulus and improving the efficiency of the protocol.

For the security of the vectors r and p in the refresh operation, their security is ensured by introducing random additive perturbations.Alice can select a random vector u of dimension n, where each element is sampled from a discrete uniform distribution $\{1, 2, \cdots, P\}$. By using $p + u$ and $r + u$, Alice successfully hides the information of r and p, making Bob's correct guessing probability $\frac{1}{p^n}$. Additionally, since the dataset dimension n is much smaller than the number of ciphertext slots $N/2$ in the CKKS scheme, Alice can choose the encryption positions of the data, while the remaining ciphertext slots are filled with random numbers. Therefore, even if Bob successfully guesses the random perturbation, he still needs to guess the positions of the data.Even if Bob manages to reconstruct the intermediate vectors r and p with a very small probability, it is still difficult for him to obtain meaningful information about the original training data.

After the training results are made public, even if one party attempts to infer the other party's data using their own data, for example, if Bob knows x_2, y_2, β and wants to infer x_1, y_1 ,we have$(A_1 + A_2)\beta = b_1 + b_2$, which means $A_2\beta - b_2 = b_1 - A_1\beta$, and it can be written in matrix form as equation (13),where a_i is a row vector of matrix A_2, $A_2 = x_2^T x_2$, $b_2 = x_2^T y_2$, and the number of samples in Bob's dataset is $md + m - d - 1$. Bob's attempt to infer Alice's dataset is at least as difficult as $2^{md+m-d-1}$, where m is typically large and $2^{md+m-d-1} \gg 2^{\lambda}$. Therefore, attempting to infer the other party's data in this scheme is extremely difficult.

On the other hand, one party can use the training results to infer the other party's dataset. This issue is generally present in two-party scenarios and is not caused by the execution process of this scheme.

$$\begin{pmatrix} \beta^T & & & \\ & \beta^T & & \\ & & \ddots & \\ & & & \beta^T \end{pmatrix} \begin{pmatrix} a_0^T \\ \vdots \\ a_n^T \\ b_2 \end{pmatrix} = b_1 - A_1\beta \tag{13}$$

5.2 Complexity Analysis

For datasets with $n - 1$ features, the logistic regression problem solved using the Newton-conjugate gradient method as described in this paper requires a total of two layers of iterations. The outer iteration is used to update the parameters of logistic regression, while the inner iteration is used to solve the Newton method for updating the direction.In the inner iteration, each round of iteration requires a matrix-vector multiplication and some inner product calculations. For the scheme described in [27], due to its encoding method, the matrix-vector multiplication requires n ciphertext multiplications, with a complexity of $O(n^2)$. If m iterations are needed, the complexity would be $O(mn^2)$.In contrast to the scheme described in [27], the matrix-ciphertext multiplication in the proposed solution described in this paper only requires one multiplication operation,

with a complexity of $O(n)$. If m iterations are needed, the complexity would be $O(mn)$.In the ciphertext domain, a single ciphertext multiplication corresponds to N polynomial multiplications. The complexity of polynomial multiplication based on the FFT(Fast Fourier Transform) is $O(N \log N)$. Therefore, the computational complexity of the ciphertext domain conjugate gradient method is $O(mn^2 N \log N)$.The CKKS homomorphic encryption scheme typically chooses a large value for N to ensure security. Therefore, compared to solving logistic regression models on plaintext data, the computational complexity on ciphertext domain significantly increases.

In the data integration phase, the original conjugate gradient method proposed in [27] requires the transmission of n ciphertexts, whereas in the proposed solution in this paper, only $4\lceil \frac{2n^2}{N} \rceil$ ciphertexts need to be transmitted. For the pre-conditioned conjugate gradient method, an additional multiplication inverse operation is required. In the scheme presented in [28], this requires the transmission of $n+3$ ciphertexts, while in the proposed solution in this paper, $8\lceil \frac{2n^2}{N} \rceil$ ciphertexts need to be transmitted. In the ciphertext domain conjugate gradient method, the modulus of the ciphertext is Q. During a single iteration, in the process of refreshing the ciphertext and computing the multiplication inverse, both parties engage in two interactions. In the scheme described in [27], a total of 7 ciphertexts need to be transmitted, including 3 fresh ciphertexts with a modulus of Q, and the remaining 4 ciphertexts have their modulus reduced to Q_0 after homomorphic computations. The communication complexity is $(0.5n+1, 5m)N \log Q + (2m-2)N \log Q_0$ bits.In the proposed solution in this paper, a total of 9 ciphertexts need to be transmitted, including 5 fresh ciphertexts with a modulus of Q, and the modulus of remaining 4 ciphertexts reduced to Q_0 after homomorphic computations. The communication complexity of this scheme is $(4\lceil \frac{2n^2}{N} \rceil + 2.5m - 2.5)N \log Q + (2m-2)N \log Q_0$ bits.

6 Experimental results

The proposed scheme is based on the SEAL library, implemented using the C++ language. The experimental environment consists of an Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz, running on the Ubuntu 22.04 operating system.

The proposed approach was evaluated on several public datasets using five-fold cross-validation. Among them, idash is the public data set of the iDash2017 homomorphic encryption track, and edin, lbw, nhanes, pcs, uis, mushroom, and splice are all standard test data sets public in the UCI database.For cryptographic schemes, the most closely related and optimal performance homomorphic encryption privacy protection schemes to the one proposed in this paper are [11][12][27].

Among them, the schemes in [11][12] are based on the CKKS homomorphic encryption scheme. They employ a fixed Hessian matrix method to compute binary logistic regression, and approximate non-linear functions using polynomial methods. However, this approach suffers from limitations in accuracy and computational overhead, making it impractical for datasets with large feature

dimensions. Our proposed scheme introduces certain communication overhead to significantly improve computational accuracy and speed.

Table 1. Implementation result 1

Dataset	Sample Num	Feature Num	Method	Outer Iter	Total Time/s			Communication/Mb			Accuracy%		
					In1	*In2*	*In5*	*In1*	*In2*	*In5*	*In1*	*In2*	*In5*
idash	1579	18	Ours	1	3.3	4.1	5.5	5.42	11.77	30.81	52.70	64.9	64.9
idash	1579	18	[27]	1	9.1	14.3	31.3	18.89	23.45	37.10	52.70	64.9	64.9
edin	1253	9	Ours	1	3.0	3.6	5.1	5.42	11.77	30.81	78.16	90.3	91.5
edin	1253	9	[27]	1	5.6	8.9	18.5	10.81	15.36	29.01	78.16	90.3	91.5
lbw	189	9	Ours	1	2.9	3.6	5.1	5.42	11.77	30.81	68.78	72.00	72.00
lbw	189	9	[27]	1	5.6	8.9	18.9	10.81	15.36	29.01	68.78	72.00	72.00
nhanes	15649	15	Ours	1	3.2	3.9	5.4	5.42	11.77	30.81	79.23	79.23	80.86
nhanes	15649	15	[27]	1	7.1	12.6	27.1	16.20	20.75	34.40	79.23	79.23	80.86
pcs	379	9	Ours	1	2.9	3.6	5.0	5.42	11.77	30.81	59.63	67.28	73.87
pcs	379	9	[27]	1	5.6	8.9	18.4	10.81	15.36	29.01	59.63	67.28	73.87
uis	575	8	Ours	1	2.8	3.5	5.1	5.42	11.77	30.81	74.43	74.43	74.43
uis	575	8	[27]	1	5.2	8.3	17.3	9.91	14.46	28.16	74.43	74.43	74.43
mushroom	8124	112	Ours	1	15.8	18.2	27.2	10.87	23.61	62.73	89.83	89.14	97.90
mushroom	8124	112	[27]	1	40.1	72.9	165.8	102.45	107.00	119.87	89.83	89.14	97.90
splice	1000	60	Ours	1	5.4	6.1	7.5	5.42	11.77	30.81	60.90	80.20	84.10
splice	1000	60	[27]	1	22.4	39.3	91.2	56.63	61.18	74.83	60.90	80.20	84.10

In [27], a homomorphic encryption-based conjugate gradient method is proposed for linear regression problems. However, the encoding method in this scheme results in a large overhead for matrix-vector ciphertext multiplication. For the ciphertext domain conjugate gradient method, we conducted detailed comparisons by setting different numbers of iterations for inner and outer layers in our experiments. The specific experimental results are shown in Tables 2, 3, and 4.

The experiments demonstrate that compared to the approach described in [27], the proposed solution in this paper exhibits advantages in both computational efficiency and communication overhead. Moreover, as the dimensionality of the dataset features increases, the advantages of the proposed solution become more pronounced.For most datasets, the number of iterations exhibits a significant variation during the first two rounds, while the subsequent iterations show less noticeable improvements in precision as the number of rounds increases. Considering the communication overhead and computational costs, it is generally sufficient to limit the number of iterations to three rounds or fewer for most datasets to meet the requirements.The experimental results indicate that the proposed solution experiences significant variations in time complexity when the feature dimension is 112. This is primarily due to the limitations imposed by the number of ciphertext slots in the CKKS scheme itself. When N is set to 2^{14}, if the dataset dimension exceeds 90 dimensions, data must be split and encoded into two ciphertexts, or alternatively, increasing the value of N to 2^{15} can also lead to significant changes in time complexity.

Table 2. Implementation result 2

Dataset	Sample Num	Feature Num	Method	Inner Iter	Total Time/s			Communication/Mb			Accuracy%		
					Out1	*Out2*	*Out5*	*Out1*	*Out2*	*Out5*	*Out1*	*Out2*	*Out5*
idash	1579	18	Ours	1	3.3	5.5	11.3	5.42	10.83	27.1	52.70	64.09	59.78
idash	1579	18	[27]	1	9.1	15.9	37.2	18.89	37.79	94.48	52.70	64.09	59.78
edin	1253	9	Ours	1	3.0	4.6	9.3	5.42	10.83	27.1	78.16	89.31	91.46
edin	1253	9	[27]	1	5.6	9.8	22.4	10.81	21.62	54.05	78.16	89.31	91.46
lbw	189	9	Ours	1	2.9	4.7	9.3	5.42	10.83	27.1	68.78	68.78	68.78
lbw	189	9	[27]	1	5.6	10.2	22.4	10.81	21.62	54.05	68.78	68.78	68.78
nhanes	15649	15	Ours	1	3.2	5.3	10.9	5.42	10.83	27.1	79.23	79.23	79.23
nhanes	15649	15	[27]	1	7.1	14.1	32.3	16.20	32.40	81.00	79.23	79.23	79.23
pcs	379	9	Ours	1	2.9	4.6	9.2	5.42	10.83	27.1	59.63	67.81	65.96
pcs	379	9	[27]	1	5.6	9.9	22.6	10.81	21.62	54.05	59.63	67.81	65.96
uis	575	8	Ours	1	2.8	4.5	9.0	5.42	10.83	27.1	74.43	74.43	74.43
uis	575	8	[27]	1	5.2	9.3	20.8	9.91	19.82	49.55	74.43	74.43	74.43
mushroom	8124	112	Ours	1	15.8	28.9	62.2	10.87	21.75	54.37	89.83	90.49	90.80
mushroom	8124	112	[27]	1	40.1	78.4	193	102.45	204.90	512.25	89.83	90.49	90.80
splice	1000	60	Ours	1	5.4	9.3	20.6	5.42	10.83	27.1	60.90	79.2	64.7
splice	1000	60	[27]	1	22.4	42.4	103.9	56.63	113.26	283.15	60.90	79.2	64.7

To further compare the communication overhead, we construct test scenarios with random datasets of different feature dimensions. Since the scenario involves horizontally distributed data, the number of samples has minimal impact on the efficiency of the proposed solution. The test dataset consists of 2000 samples. Figure 2 compares the communication overhead of different schemes for varying feature dimensions.Figure 3 compares the communication overhead of different numbers of inner and outer iteration rounds for a test dataset with 9 features and 2000 samples.Figure 4 compares the communication overhead of different numbers of inner and outer iteration rounds for a test dataset with 90 features and 2000 samples.

Additionally, experiments demonstrate that by setting the number of outer iteration rounds to 1, the communication overhead of the proposed solution remains lower than that of the scheme described in [27] for the first k rounds, where k is proportional to the feature dimension n. Due to the fast convergence of the Newton method, most datasets require fewer than 3 rounds to meet the requirements, giving the proposed solution an advantage. On the other hand, by setting the number of inner iteration rounds to 1, as the number of outer iteration rounds increases, the communication overhead of the proposed solution is superior to that of the scheme in [27], and this advantage becomes more pronounced with the increase in feature dimension.

Finally, for the privacy-preserving logistic regression scheme, comparisons were also made with the solutions proposed in [11][12][22]. The solution proposed in [11][12] adopts a fixed Hessian matrix approach for computing binary logistic regression, while the solution in [22] addresses the problem of multi-class logistic regression using gradient descent. For non-linear functions, these approaches all employ polynomial approximation methods, leading to limitations in accuracy and computational overhead,making them impractical for datasets

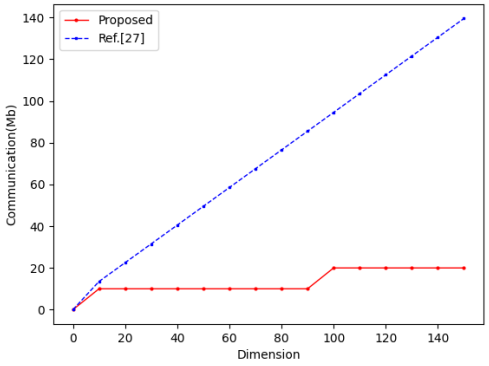

Fig. 2. Comparison of communication 1.

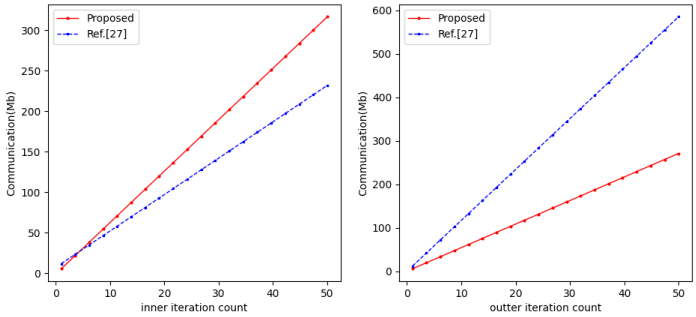

Fig. 3. Comparison of communication 2.

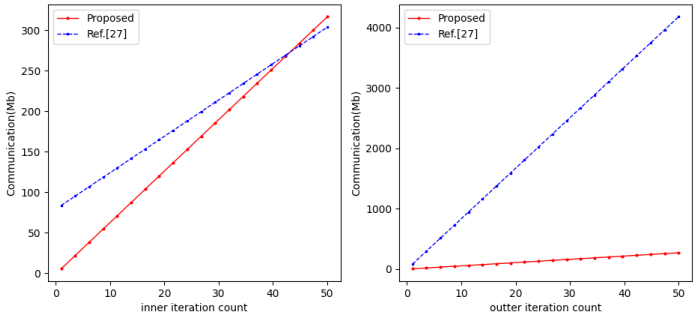

Fig. 4. Comparison of communication 3.

with large feature dimensions.In contrast, the proposed solution introduces a certain communication overhead to significantly improve computational accuracy and speed. By utilizing SIMD technology and OvR strategy, it can be extended to multi-class scenarios. For specific experimental results, refer to Table 3.

Table 3. Implementation result 3

Dataset	ClassNum	Method	Iter	Time	Accuracy
idash	2	Ours	1	3.3s	52.70%
idash	2	Ours	2	4.1s	64.9%
idash	2	[11]	3	3.61min	53.38%
idash	2	[12]	7	6.07min	62.87%
edin	2	Ours	1	3.0s	78.16%
edin	2	Ours	1	3.3s	52.70%
edin	2	[11]	3	0.5min	84.4%
edin	2	[12]	7	3.6min	91.4%
lbw	2	Ours	1	2.9s	68.78%
lbw	2	Ours	2	3.6s	72.00
lbw	2	[11]	3	0.4min	68.65%
lbw	2	[12]	7	3.3min	69.19%
nhanes	2	Ours	1	3.2s	79.23%
nhanes	2	Ours	2	3.9s	79.23%
nhanes	2	[11]	3	3.7min	79.22%
nhanes	2	[12]	7	7.7min	79.22%
pcs	2	Ours	1	2.9s	59.63%
pcs	2	Ours	2	3.9s	67.28%
pcs	2	[11]	3	0.6min	64%
pcs	2	[12]	7	3.5min	68.27%
uis	2	Ours	1	2.8s	74.43%
uis	2	Ours	2	3.5s	74.43%
uis	2	[11]	3	0.5min	74.43%
uis	2	[12]	7	3.5min	74.44%
iris	3	Ours	6	5.3S	74.00%
iris	3	Ours	8	8.9s	82.00 %
iris	3	[22]	7	13.51min	76.67%

7 Conclusions and future work

This paper combines the CKKS homomorphic encryption technique to design a ciphertext domain Newton-Conjugate Gradient Logistic Regression algorithm suitable for scenarios where two users participate and the dataset exhibits a horizontally distributed pattern.

By introducing a small amount of interaction, the solution improves computational efficiency, resulting in a significant reduction in time complexity. For the majority of datasets, the proposed solution achieves comparable accuracy to existing privacy-preserving schemes with 5 to 7 rounds of iterations, while limiting the number of iterations to 3 rounds or fewer. Additionally, it efficiently operates on datasets with high feature dimensions. Furthermore, this paper leverages

SIMD technology and OvR strategy to simultaneously train multiple binary classification models, binary logistic regression is extended to a multi-class scenario. However, there is still room for improvement in terms of algorithm accuracy. In future work, we plan to explore alternative strategies such as softmax regression to further extend the binary logistic regression problem to a multi-class scenario.

Acknowledgments. This study was funded by National Key Research and Development Program of China (2020YFA0712300); Chongqing's Leading Academician-Led Special Project for Technology Innovation Guidance (2022YSZX-JCX0011CSTB, cstc2021yszx-jcyjX0004, CSTB2023YSZX-JCX0008,cstc2021jcyj-msxmX0821).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article

References

1. Fang, Haokun, and Quan Qian. "Privacy preserving machine learning with homomorphic encryption and federated learning." Future Internet 13.4 (2021): 94.
2. Park, Jaehyoung, Nam Yul Yu, and Hyuk Lim. "Privacy-Preserving Federated Learning Using Homomorphic Encryption With Different Encryption Keys." 2022 13th International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2022.
3. Yang Yuejia, Hua Bei, Zhong Zhiwei, Gao Mi. "Collaborative Computing of Privacy-Preserving Logistic Regression Based on Homomorphic Encryption." Computer Engineering, 2023, 49(4): 23-31.
4. Ameur, Yulliwas, Samia Bouzefrane, and Vincent Audigier. "Application of homomorphic encryption in machine learning." Emerging Trends in Cybersecurity Applications. Cham: Springer International Publishing, 2022. 391-410.
5. Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21.2 (1978): 120-126.
6. Shi, Haoyi, et al. "Secure multi-pArty computation grid LOgistic REgression (SMAC-GLORE)." BMC medical informatics and decision making 16 (2016): 175-187.
7. Xie, Wei, et al. "Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers." arXiv preprint arXiv:1611.01170 (2016).
8. Fan, Yongkai, et al. "Privacy preserving based logistic regression on big data." Journal of network and computer applications 171 (2020): 102769.
9. Kim, Andrey, et al. "Logistic regression model training based on the approximate homomorphic encryption." BMC medical genomics 11.4 (2018): 23-31.
10. Carpov, Sergiu, et al. "Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption." Cryptology ePrint Archive (2019).
11. Han, Kyoohyung, et al. "Logistic regression on homomorphic encrypted data at scale." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.
12. Chiang, John. "Privacy-preserving logistic regression training with a faster gradient variant." arXiv preprint arXiv:2201.10838 (2022).
13. De Cock, Martine, et al. "High performance logistic regression for privacy-preserving genome analysis." BMC Medical Genomics 14 (2021): 1-18.

14. Kim, Miran, et al. "Secure and differentially private logistic regression for horizontally distributed data." IEEE Transactions on Information Forensics and Security 15 (2019): 695-710.
15. Yu, Xiaopeng, et al. "Privacy-Preserving Vertical Collaborative Logistic Regression without Trusted Third-Party Coordinator." Security and Communication Networks 2022 (2022).
16. He, Haoran, et al. "A privacy-preserving decentralized credit scoring method based on multi-party information." Decision Support Systems 166 (2023): 113910.
17. Sun, Huizhong, et al. "Privacy-preserving vertical federated logistic regression without trusted third-party coordinator." Proceedings of the 2022 6th International Conference on Machine Learning and Soft Computing. 2022.
18. He, Daojing, et al. "Secure logistic regression for vertical federated learning." IEEE Internet Computing 26.2 (2021): 61-68.
19. Mohassel, Payman, and Yupeng Zhang. "Secureml: A system for scalable privacy-preserving machine learning." 2017 IEEE symposium on security and privacy (SP). IEEE, 2017.
20. Ghavamipour, Ali Reza, Fatih Turkmen, and Xiaoqian Jiang. "Privacy-preserving logistic regression with secret sharing." BMC Medical Informatics and Decision Making 22.1 (2022): 1-11.
21. Liu, Yi, et al. "Secure multi-label data classification in cloud by additionally homomorphic encryption." Information Sciences 468 (2018): 89-102.
22. Xu X W, Cai B, Xiang H, Sang J. "Multinomial Logistic Regression Model Based on Homomorphic Encryption. Journal of Cryptologic Research." 2020, 7(2): 179-186
23. Sarkar, Esha, et al. "Privacy-preserving cancer type prediction with homomorphic encryption." Scientific reports 13.1 (2023): 1661.
24. Zaidi, Nayyar A., and Geoffrey I. Webb. "A fast trust-region newton method for softmax logistic regression." Proceedings of the 2017 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2017.
25. Yang, Jianke. "Newton-conjugate-gradient methods for solitary wave computations." Journal of Computational Physics 228.18 (2009): 7007-7024.
26. Nazareth, John L. "Conjugate gradient method." Wiley Interdisciplinary Reviews: Computational Statistics 1.3 (2009): 348-353.
27. Lyu Y, Wu W Y. "Two-party Privacy-preserving Ridge Regression Scheme with Applications. Journal of Cryptologic Research." 2023, 10(2): 276-288
28. Cheon, Jung Hee, et al. "Homomorphic encryption for arithmetic of approximate numbers." Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. Springer International Publishing, 2017.
29. Gentry, Craig, Shai Halevi, and Nigel P. Smart. "Homomorphic evaluation of the AES circuit." Annual Cryptology Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
30. Lindner, Richard, and Chris Peikert. "Better key sizes (and attacks) for LWE-based encryption." Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings. Springer Berlin Heidelberg, 2011.
31. Lyubashevsky, Vadim, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings." Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29. Springer Berlin Heidelberg, 2010.