RESEARCH

Privacy-preserving Logistic Regression with Secret Sharing

Ali Reza Ghavamipour[1*], Fatih Turkmen[1] and Xiaoqian Jiang[2]

Abstract

Background: Logistic regression (LR) is a widely used classification method for modeling binary outcomes in many medical data classification tasks. Research that collects and combines datasets from various data custodians and jurisdictions can excessively benefit from the increased statistical power to support their analyzing goals. However, combining data from these various sources creates significant privacy concerns that need to be addressed.

Methods: In this paper, we proposed secret sharing-based privacy-preserving logistic regression protocols using the Newton-Raphson method. Our proposed approaches are based on secure Multi-Party Computation (MPC) with different security settings to analyze data owned by several data holders.

Results: We conducted experiments on both synthetic data and real-world datasets and compared the efficiency and accuracy of them with those of an ordinary logistic regression model. Experimental results demonstrate that the proposed protocols are highly efficient and accurate.

Conclusions: This study introduces iterative algorithms to simplify the federated training a logistic regression model in a privacy-preserving manner. Our implementation results show that our improved method can handle large datasets used in securely training a logistic regression from multiple sources.

Keywords: logistic regression; secret sharing; multi-party computation; privacy-preserving; newton-raphson

1 Background

Nowadays, patient data (i.e., medical records and genomics) are rapidly being collected worldwide, which has expanded in volume faster than anyone predicted. Medical data analysis often benefits from collecting more samples to increase the power of statistical analysis and the robustness of machine learning models. To acquire more valuable information for research, hospitals and research centers would like to collaborate by sharing their data and findings in a central location. The main benefits of the collaborative sharing and processing of data are more accurate detection and diagnosis of disorders, prediction of disease origin, and development of drugs.

Various data analytic techniques can be employed to extract information from the shared datasets. The logistic regression model [1], one of the most popular prediction models, is now widely used in medical research. By analyzing data, logistic regression estimates a particular event's probability based on previously observed data and creates a binary classification model. More specifically, the value of a binary variable is predicted based on several independent variables. For example, logistic regression could develop a model suitable for identifying malignant breast cancers based on tumor size, patient age, blood type, and genetic inheritance [2]. Statistical models need a sufficient sample size to ensure necessary statistical power in data analysis, and machine learning models require even a large sample size (due to the consideration of interactions and non-linearity assumptions) [3]. It is also beneficial to combine and compare data from different sources (i.e., ensuring generalizability). However, collecting data from multiple sources often lead to privacy concerns. Due to institutional policies and legal restrictions, hospitals and medical centers often have concerns about sharing their sensitive data (i.e., especially patient-level information) with other institutions. Therefore, it is essential to use privacy-preserving solutions and techniques, such as secure Multi-Party Computation (MPC), to train a logistic regression model over shared data between multiple data holders [4].

*Correspondence: a.r.ghavamipour@rug.nl
[1]University of Groningen, Nijenborgh 9, Groningen, Netherlands
Full list of author information is available at the end of the article

In this paper, our particular focus is on enabling logistic regression between multiple data holders, such as hospitals and research centers, in a privacy-preserving manner. With the proposed protocols, multiple parties can train a logistic regression model using the Newton-Raphson optimization method under different security assumptions. Thus, the main contributions of this paper include:

- A novel privacy-preserving algorithm for computing logistic regression models that is very accurate but not as efficient, although acceptable.
- A second algorithm that is very efficient but less accurate due to the use of multiple approximations.
- Implementation of the proposed algorithms in both honest and dishonest majority security settings.
- Evaluation of the proposed protocols on various real-world and generated synthetic datasets.

We consider a setting in which data are horizontally partitioned, which indicates that data holders have precisely the same variables but different values for those variables.

2 Related Work

Various approaches make use of cryptographic techniques such as multi-party computation (MPC), homomorphic encryption, and differential privacy. Previous studies showed the practicability of building a secure distributed logistic regression across multiple data holders. However, due to the complexity of the underlying secure computation primitives, their methods suffer from multiple issues such as scalability, accuracy loss, and low efficiency.

The Grid Binary LOgistic REgression (GLORE) model was proposed by Wu et al. [5] to support privacy-preserving logistic regression in a distributed manner. GLORE estimates global model parameters for horizontally partitioned data without necessarily sharing patients. Rather than directly sharing sensitive data with other institutes, the decomposable intermediary results with significantly less sensitive information are transferred to build a global training protocol for logistic regression. However, in their proposed methods, sensitive data could leak due to disclosure of the information matrix and summary statistics.

Shi et al. [5] proposed a secure multi-party computation framework for grid logistic regression (SMAC-GLORE), which protects the patient data's confidentiality and privacy. SMAC-GLORE preserves the intermediary results based on garbled circuits during iterative model learning. Various approaches such as secure matrix multiplication and addition and fixed-Hessian methods have been employed to estimate the maximum likelihood. However, due to the garbled circuit constraints, SMAC-GLORE cannot handle a large number of records and features. Also, it uses the Taylor series approximation approach to evaluate the sigmoid function, which causes accuracy loss in the final result.

Xie et al. [6] developed PrivLogit, which performs distributed privacy-preserving logistic regression and uses Yao's garbled circuits and Paillier encryption. PrivLogit needs the data owners to perform computations on their data before encryption to compute parts of a logistic regression matrix. However, their proposed method requires an expensive computational cost to calculate the intermediate results.

Cock et al.[7] proposed an information-theoretically privacy-preserving model training protocols based on secret sharing-based building blocks such as distributed multiplication, distributed comparison, bit-decomposition of shares. Similar SecureML, their proposed protocol requires multiplication triples distributed during a setup phase with or without a trusted authority. Unlike SecureML, which is secure in the computational context, they engage in the information-theoretic model using secret sharing-based MPC and employ commodity-based cryptography [8] to decrease the number of communications.

SecureML [9] was one of the fastest protocols for privacy-preserving logistic regression models training based on secure MPC. The SecureML protocol is divided into an offline (to generate and distributing multiplication triples) and an online phase. Their proposed multiplication protocol is based on a straightforward and efficient security setting introduced by Beaver [8]. Also, to compute the activation functions, they proposed a new comparison-based activation function that converges to 0 and 1. Unlike our work that employs the Newton Raphson optimization method, SecureML focuses on the mini-batch gradient descent.

Other than MPC-based solutions, two popular methods have been considered. The first one is homomorphic encryption [10], which allows for computation to be performed over encrypted data, and has been applied to privacy-preserving logistic regression [11][12][13][14][15][16][17]. In most of these works, polynomial approximations need to be made to evaluate non-linear functions in machine-learning algorithms. The second method is differential privacy, a universally accepted mathematical structure for protecting data privacy. The main application of differential privacy in machine learning is when the model is published publicly after training in a way that personal data points cannot be distinguished from the released model [18][19][20][21].

3 Preliminaries

3.1 Secure multi-party computation

Secure Multi-Party Computation (MPC) allows computation parties to compute an arbitrarily agreed function of their private inputs. During the computation, no party should reveal its private inputs to the other parties or any third party. This is formalized as a secure function evaluation where n parties compute a function

$$f(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_n) \tag{1}$$

such that each computation party P_i, where $1 \leq i \leq n$, provides its input x_i and learns its output y_i. The secret shared value can be exposed to each party by combining with other parties shares. Since our MPC solution is based on additive secret sharing, we briefly discuss this concept in the next section.

3.2 Secret sharing

Secret sharing is a set of techniques that allows a secret value x to be distributed among n participants as $x_1, ..., x_n$ so that each party P_i receives a random share x_i $(mod\ p)$ over some prime p [1]. In secret sharing-based secure computation schemes, a different number of sensitive data holders (input parties) can secretly share their data among other participants. In this paper, we use the n-out-of-n additive secret sharing scheme. In this scheme, an integer u additively shares between n participants. In other words, each input party pick $n-1$ randomly generated values and sends them to all other participants. Also, one of the parties is provided by the secret u minus the sum of those randomly generated values, which permits the reconstruction of the original value by summing all of the shares.

In what follows, we will use $[\![x]\!]$ to denote secret shares that reconstruct to x. A share $[\![x]\!]$ is an n-tuple with each computing party holding precisely one element of the tuple and $[\![x]\!]_i$ denotes the share held by the i_{th} party.

3.3 Security model

We consider a set of input parties who aim to train a logistic regression model on their sensitive data. We assume the data is distributed horizontally among the input parties, where each independent database contains only a sub-population. Furthermore, we consider a set of computation parties n that receives secret-shared data in a setup phase (this phase is not considered in any future computation).

[1] the modular notation is dropped for the means of conciseness and ease of composition

We assume the computation parties are non-colluding (the servers cannot be controlled by one authority), independent (if the adversary controls one party, other parties behave honestly), and honest but curious (each party correctly follows the protocol but might be curious about the information transferred between other parties). However, one or more computation parties may get corrupted, and these corrupted parties could involve more than half of the computation parties. We propose logistic regression training protocols for both honest majority and dishonest majority assumptions in this paper. In the case with the honest majority assumption, the adversary may actively corrupt $t < n/2$ players. We implement this protocol in three-party settings where only one party can be corrupted at most. In the second protocol, we propose a protocol in which the number of corrupted parties could be more than $n/2$. To achieve the highest efficiency, we implement this protocol in the two-party setting.

3.4 Addition and multiplication

Various operations can be performed on secret shared data through tailored protocols. Based on the considered security models mentioned in section 3.3, we employ addition and multiplication as the key operations in our work. Under both introduced security models, the addition of two secrets can be performed locally without any communication: $[\![x]\!]+[\![y]\!] = ([\![x]\!]_1 + [\![y]\!]_1, [\![x]\!]_2 + [\![y]\!]_2, \ldots, [\![x]\!]_n + [\![y]\!]_n)$. However, the multiplication of additively secret shared values requires network communication.

In the honest majority setting, we use the multiplication protocol proposed by Bogdanov et al. [22] (classical approach). Their protocol is based on the following equation:

$$\begin{aligned} xy = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) = \\ (x_1y_1 + x_1y_3 + x_3y_1) + \\ (x_2y_2 + x_2y_1 + x_1y_2) + \\ (x_3y_3 + x_3y_2 + x_2y_3) \\ = \sum_{i=1}^{3}\sum_{j=1}^{3} x_iy_j \end{aligned} \tag{2}$$

As shown in Equation 2, to perform the multiplication operation, each computation party requires its adjacent computation party's input share. However, Bogdanov et al. stated that sending the input share to the nearby computation party may reveal more information about the input data than necessary. To solve this issue, they introduced a re-sharing protocol to distribute a new share of the original value in each computation party at the beginning and end of the multiplication operation.

In the dishonest majority setting, we use the Beaver triples technique [23] to perform multiplication. This method requires a trusted initializer to pre-distribute the multiplication triples (random and independent) shares ($[\![a]\!]$, $[\![b]\!]$, $[\![c]\!]$) between the computation parties in a way that $a.b = c$. To perform the multiplication, each computation party computes $[\![d]\!] = [\![x]\!]$ - $[\![a]\!]$ and $[\![e]\!] = [\![y]\!]$ - $[\![b]\!]$ locally and then reveals both $[\![d]\!]$ and $[\![e]\!]$. Revealing these shares does not compromise the security of sensitive data, as $[\![a]\!]$ and $[\![b]\!]$ have been randomly generated and thus mask the secret values. Next, each party locally computes:

$$[\![w]\!]_i = [\![c]\!]_i + [\![e]\!].[\![b]\!]_i + [\![d]\!].[\![a]\!]_i + [\![e]\!].[\![d]\!]$$

where $[\![w]\!]_i$ is a share of the result of the multiplication.

Note that after distributing the multiplicative triples needed, the trusted initializer is not involved in the rest of the protocol. Thus, the trusted initializer does not understand the function to be computed or the computation inputs.

3.5 Inversion

As we will see in Section 4, we will need the operations of inversion and matrix inversion in order to implement logistic regression. The MPCs protocols discussed previously support only addition and multiplication. Since the accurate implementation of inversion significantly increases the computational cost, we use the approximation method introduced by Nardi [24]. Nardi's method converts the matrix inversion problem into an iterative procedure of matrix multiplication and addition. In this method, we look for a matrix X to find the inversion of the matrix B according to:

$$X^{-1} = B$$

The main idea is to define a function of which matrix X represents its root ($f(X) = 0$). Therefore, the function $f(x)$ is defined as follows:

$$f = X^{-1} - B$$

To find the root of the function f , Nardi suggested using the Newton-Raphson method. By applying this method (take the derivative of function $f(X)$ and applying the general iterative Newton method), a stable numerical iterative approximation takes the following form:

$$\begin{aligned} X_{s+1} &= 2X_s - X_s M_s \quad X_0 = c^{-1}\mathbb{I} \\ M_{s+1} &= 2M_s - M_s^2 \quad M_0 = c^{-1}B \end{aligned} \tag{3}$$

where X_0 and M_0 are the initial guesses, $\mathbb{I}$ is an identity matrix, and c is a constant. After convergence, X_s contains an approximation of matrix B's inversion.

3.6 Logistic regression

Logistic regression is a standard machine learning technique that is commonly used in various areas of research. It predicts the probability that a dependent variable belongs to a class. This paper will consider the binary classification, where the dependent variable belongs to two possible classes. The logistic model is intended to describe a probability, which is always a number between 0 and 1.

Assume a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ of n records, where x_i is the m-dimensional feature vector of each record and the y_i is a vector of labeled binary outcomes. The logistic regression model is given by:

$$P(y_i = 1|x; \beta) = \pi(\beta^T x_i) = \frac{1}{1 - e^{-\beta^T x_i}} \tag{4}$$

where $\beta = (\beta_0, \ldots, \beta_d)$ is p-dimensional regression coefficients vector, y is the observation of binary responses, and x is the feature vectors. The purpose of using this method is to obtain the parameter vector β that maximizes the log-likelihood function:

$$l(\beta) = -\sum_{i=1}^{n} \log(1 + e^{-\beta^T x_i}) \tag{5}$$

By determining the parameters β, the classifier can predict the class label of new feature vectors.

4 Methods

4.1 Estimating model coefficients

Since logistic regression cannot be found in a closed form, model estimation is often accomplished by an iterative optimization over the log-likelihood function. Newton-Raphson [25] is a numerical iterative method that eventually approaches the optimal values of the β coefficients. For each iteration, the coefficient estimates are updated by:

$$\beta_{new} = \beta_{old} - \mathbb{H}^{-1}(\beta_{old})\nabla(\beta_{old}) \tag{6}$$

where ∇ and $\mathbb{H}$ respectively correspond to the gradient and the Hessian of the log-likelihood function

evaluated the current estimate of the β coefficients. The gradient and Hessian for logistic regression can be computed as follows:

$$\nabla(\beta) = \frac{\partial l(\beta)}{\partial \beta} = \mathbb{X}^{\mathbb{T}}(y - \pi) \tag{7}$$

$$\mathbb{H}(\beta) = \frac{\partial^2 f}{\partial \beta \partial \beta^{\mathbb{T}}} = \mathbb{X}^{\mathbb{T}}\mathbb{W}\mathbb{X} \tag{8}$$

where $\mathbb{W}$ is a diagonal matrix with elements defined as $a_{i,i} = \pi(1-\pi)$ and π is the vector of probabilities.

4.2 Gradient

To compute the gradient (7), first we need to compute the Sigmoid function (π). To do this, we consider two different cases in our paper:

$$\pi(z) = \frac{1}{1+e^{-z}} = (1+e^{-z})^{-1} \tag{9}$$

a) Computation of the exact value of the Sigmoid function: The main challenges of computing the exact value of the Sigmoid function vector are the performing of exponentiation, inversion, and matrix inversion. To perform inversion and matrix inversion, we use the solutions discussed in Section 3.5. However, performing exponentiation by the considered secret sharing techniques is still challenging.

To tackle this issue, we implement exponentiation by using addition and multiplication. The solution is based on the idea of using additive secret sharing one more time. In other words, each computation party plays the role of a sensitive data holder. To do this, each computation party i computes $e^{[\![z_i]\!]}$ locally and then computes n different shares of it (n indicates the total number of computation parties) and sends each of these values to the correspondence computation parties. After each computation party received the other computation parties' share of $e^{[\![z_i]\!]}$, by using the MPC multiplication operation, it computes $([\![e^{[\![z_1]\!]}]\!]_1 * [\![e^{[\![z_2]\!]}]\!]_2 * \cdots * [\![e^{[\![z_i]\!]}]\!]_n)$ which is equal to $([\![e^{[\![z_1]\!]+[\![z_2]\!]+\cdots+[\![z_i]\!]}]\!])$. Therefore, each computation party has a valid share of $[\![e^{[\![z]\!]}]\!]$ and using the addition and inversion operation, the exact value of the Sigmoid function will be computed.

b) Least Squares Approximation of the Sigmoid Function: The introduced method to compute the sigmoid function's exact value might have scalability issues due to the substantial required number of multiplications. In order to improve the performance, we use the least-squares approximation of the sigmoid function over the interval [-8,8] introduced by Kim et al. [26]. We adapt this approximation method and consider the degree 3, 5, and 7 least-squares polynomials:

$$\begin{cases} g_3(x) = 0.5 + 1.20096.(x/8) - 0.81562.(x/8)^3 \\ g_5(x) = 0.5 + 1.53048.(x/8) - 2.3533056.(x/8)^3 \\ \qquad +1.3511295.(x/8)^5 \\ g_7(x) = 0.5 + 1.73496.(x/8) - 4.19407.(x/8)^3 \\ \qquad +5.43402.(x/8)^5 - 2.50739.(x/8)^7 \end{cases}$$

The degree 3 least-squares approximation requires fewer multiplications, while the degree 7 polynomial has more immeasurable precision.

4.3 Hessian

The Hessian matrix $\mathbb{H}$ denotes the second partial derivatives of the maximum likelihood function $l(\beta)$. In every iteration, the Hessian matrix has to be updated by the newest β, and its inversion has to be computed. To evaluate the Hessian matrix, we can consider two different methods. First, we can compute the exact value of the Hessian matrix by performing the required MPC-based multiplication. However, the exact evaluation of the Hessian matrix is considerably expensive in computational terms. To solve this issue, we approximate the Hessian matrix with a fixed matrix instead of updating it in every iteration. More specifically, we can replace the fixed Hessian matrix with its approximation $\tilde{\mathbb{H}}$ (Equation 10) that only needs to be computed and inverted once.

$$\tilde{\mathbb{H}} = \frac{-1}{4} X X^T \tag{10}$$

Böhning [27] has proved that if $\tilde{\mathbb{H}} - \mathbb{H}$ is positive definite and $\mathbb{H} \leq H$ then the convergence of this method is guaranteed. Also, because $\tilde{\mathbb{H}}$ does not depend on β, we can pre-compute the Hessian and its inverse one time and used it in all iterations.

4.4 Privacy-preserving Logistic Regression Training

In this work, we assumed that the result party desires to compute the logistic regression model over collected data by different data owners. Each data owner computes multiple shares (based on the number of computation parties) of its sensitive data and separately sends them to each computation party. Note that each computation party receives an equal number of dependent X_i and independent y_i variables. Each computation party should append the received shares and their corresponding dependent variables in the correct order. Finally, computation parties send their computed

shares of logistic regression coefficient to the result party, and the result party simply sums these shares together to compute the final result.

We now present our privacy-preserving logistic regression training algorithms that employ the previously mentioned approaches. These algorithms summarize the crucial steps in the proposed protocols for both honest and dishonest majority security assumptions. In our proposed algorithms, each data owner provides a share of data for the computation parties as input. The only output of the algorithm is the computed model coefficients β. Also, to prevent unnecessarily revealing information about the input, we will not employ a convergence check after each iteration. n_{iter} specifies the upper bound of the number of iterations needed for convergence.

Algorithm 1 Accurate Logistic Regression Training algorithm

Input: A share of input data ($[\![X]\!]$, $[\![y]\!]$) from data owners, n_{iter} number of iterations
Output A share of computed vector of coefficient $[\![\beta]\!]$

1: **procedure**
2: $[\![\beta]\!], [\![\beta_{old}]\!] \leftarrow 0$
3: $[\![\tilde{\mathbb{H}}]\!] \leftarrow \frac{-1}{4}[\![X]\!].[\![X]\!]^T$
4: $[\![\tilde{\mathbb{H}}]\!] \leftarrow [\![\tilde{\mathbb{H}}]\!]^{-1}$
5: **for** 0 to n_{iter} **do**
6: $[\![\beta]\!] = [\![\beta_{old}]\!]$
7: $z \leftarrow [\![X]\!].[\![\beta]\!]$
8: **procedure** COMPUTING $[\![\pi]\!]$
9: locally computes $e^{[\![z]\!]}$
10: send/receive a share of $e^{[\![z]\!]}$ to/from other CPs
11: $[\![e^z]\!] = [\![e^{[\![z]\!]}]\!]_1.[\![e^{[\![z]\!]}]\!]_2 ... [\![e^{[\![z]\!]}]\!]_i$
12: $[\![\pi]\!] = [\![(1+e^z)]\!]^{-1}$
13: **end procedure**
14: $[\![\nabla(\beta_{temp})]\!] = [\![y]\!] - [\![\pi]\!]$
15: $[\![\nabla(\beta)]\!] = [\![X]\!]^T.[\![\nabla(\beta_{temp})]\!]$
16: $[\![\beta_{old}]\!] = [\![\beta]\!] - [\![\tilde{\mathbb{H}}^{-1}]\!].[\![\nabla(\beta)]\!]$
17: **end for**
18: Return $[\![\beta]\!]$
19: **end procedure**

In Algorithm 1, we propose a very accurate privacy-preserving logistic regression model training protocol. In this algorithm, we only employ highly accurate approximations such as matrix inversion and fixed Hessian, which have a negligible effect on the computation output's accuracy. Moreover, instead of approximating the Sigmoid function, we use our introduced approach in section 4.2 to compute the Sigmoid function's exact value (lines 8-12).

The main purpose of proposing Algorithm 2 is to achieve a highly efficient privacy-preserving logistic regression model training protocol. Various approximation approaches such as fixed Hessian matrix, least-square approximation for the Sigmoid function, and matrix inversion algorithm are employed to obtain our goal. This logistic regression training algorithm demonstrates how the introduced approximation approaches can be efficiently combined to compute the logistic regression coefficient in a privacy-preserving manner.

Algorithm 2 Approximation-based Logistic Regression Training algorithm

Input: A share of input data ($[\![X]\!]$, $[\![y]\!]$) from data owners, n_{iter} number of iterations
Output A share of computed vector of coefficient $[\![\beta]\!]$

1: **procedure**
2: $[\![\beta]\!], [\![\beta_{old}]\!] \leftarrow 0$
3: $[\![\tilde{\mathbb{H}}]\!] \leftarrow \frac{-1}{4}[\![X]\!].[\![X]\!]^T$
4: $[\![\tilde{\mathbb{H}}]\!] \leftarrow [\![\tilde{\mathbb{H}}]\!]^{-1}$
5: **for** 0 to n_{iter} **do**
6: $[\![\beta]\!] = [\![\beta_{old}]\!]$
7: *Compute* $[\![\pi]\!]$
8: $[\![\nabla(\beta_{temp}) = [\![y]\!] - [\![\pi]\!]$
9: $[\![\nabla(\beta)]\!] = [\![X]\!]^T.[\![\nabla(\beta_{temp})]\!]$
10: $[\![\beta_{old}]\!] = [\![\beta]\!] - [\![\tilde{\mathbb{H}}^{-1}]\!].[\![\nabla(\beta)]\!]$
11: **end for**
12: Return $[\![\beta]\!]$
13: **end procedure**

5 Results

In this section, we first describe computational efficiency evaluations in terms of CPU time and memory consumption for the proposed algorithms over a real-world dataset and generated synthetic data. We finally describe the accuracy evaluations of these protocols and theoretically discuss the communication cost.

5.1 Implementation Details

We implemented both of the algorithms with two introduced security settings for MPC in section 3 in Python. Algorithm 1 is implemented using Beaver triple-based MPC (Accurate BMPC) and classical MPC (Accurate CMPC). The Beaver triple-based version of algorithm 2 is called BMPC, and the classical-based MPC implementation of this algorithm is called CMPC. Moreover, to have a decent comparison, the ordinary logistic regression (OLR), which does not use MPC, is implemented.

Experiments were performed on an ARM-based M1 processor with 16GB memory, running a macOS operation system. Also, to eliminate the impact of network latency, we simulated the (distributed) computing nodes on a single computer with multiple threads. Each experiment was performed at least 10 times and reported the mean of the output. During the validation, we employed both synthetic data and real-world data sets.

We report the evaluation results concerning **computational efficiency** in terms of CPU time and memory consumption and **result accuracy**. For a fair comparison on the efficiency, we used four real-world

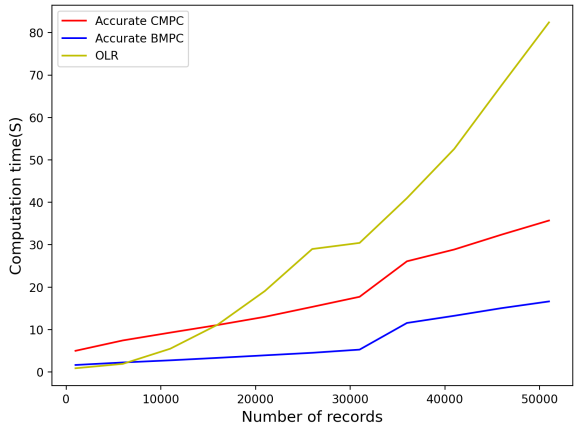

Figure 1: Efficiency comparison for increasing number of records using accurate algorithm 1

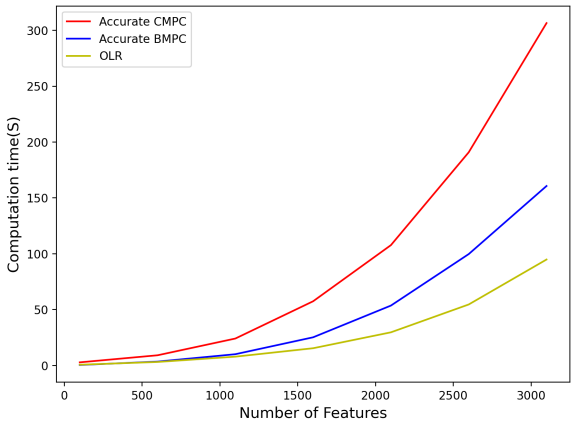

Figure 2: Efficiency comparison for increasing number of features using accurate algorithm 1

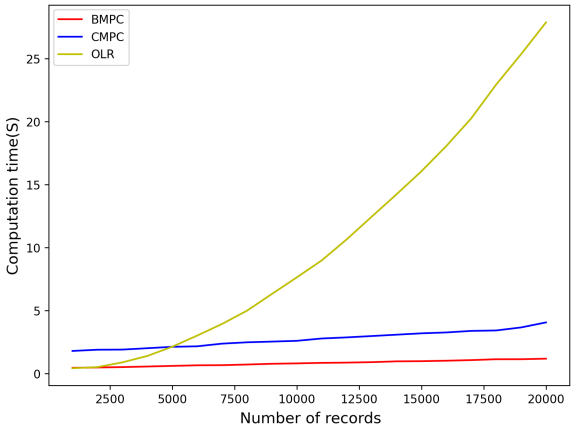

Figure 3: Efficiency comparison for increasing number of records using approximation-based algorithm 2

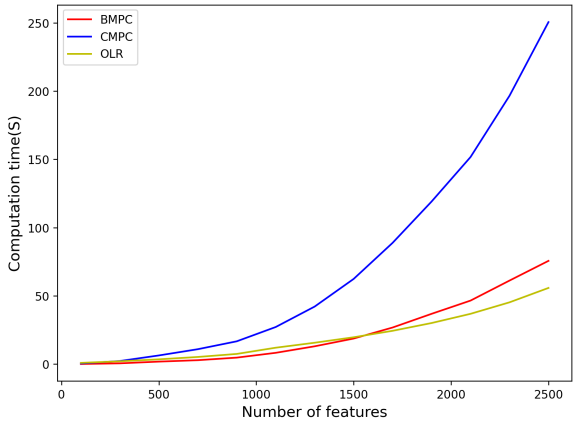

Figure 4: Efficiency comparison for increasing number of features using approximation-based algorithm 2

data sets: Pima Indians Diabetes Dataset (PIMA) [28], Low Birth Weight Study (LBW) [29], Prostate Cancer Study (PCS) [30], and Umaru Impact Study datasets (UIS)[31]. All datasets have a single binary outcome variable. Also, to satisfy the demand for large-scale studies between multiple research institutions with a large number of records, we examine our protocols with synthetic data sets of varying sizes. We generated synthetic data consists of up to 1 million records spanning up to 3000 features representing most real-world use cases.

5.2 Efficiency

To compare our protocols' efficiency with an ordinary logistic regression, first, we measure the CPU time of our protocol when the number of features is constant (i.e., 250) and the number of records increases. We then calculate the CPU time of the protocols when the number of records is fixed (i.e., 7000) and the number of features increases.

The CPU time of the proposed protocols is heavily influenced by the number of records and features of the training set. Figures 1 and 2 illustrate the CPU time of those implemented based on algorithm 1 (Accurate BMPC and Accurate CMPC). As is shown in Figure 1, Accurate BMPC has the best results when the number of records increases. This protocol computes a logistic regression model over a train set with 50000 records and 500 features in less than 15 seconds which is 20 seconds faster than Accurate CMPC protocol and 70 seconds faster than OLR. Nevertheless, increasing the number of features has a higher impact than OLR. As is shown in Figure 2, by using the OLR protocol, a logistic regression model can be trained over a training set with 7000 records and 3000 features in around 90 seconds. However, computing this model using Accu-

Table 1: Memory consumption comparison of our proposed protocols using generated synthetic datasets

Records Number	Feature Number	OLR	BMPC	CMPC	Accurate BMPC	Accurate CMPC
10000	50	3666	1325	688	1316	888
20000	70	12559	1717	1016	1665	1325
30000	90	22736	2743	1702	2654	2033
40000	100	29110	3775	2337	3622	2672
50000	100	34394	4575	2878	4406	3220
50000	200	34512	8516	5437	8013	5770

Table 2: A comparison between model parameters β learned using the proposed protocols and ordinary logistic regression protocol over LBW dataset

β	Ordinary LR	Accurate BMPC	BMPC			Accurate CMPC	CMPC		
			3	5	7		3	5	7
β_1	0.01574	0.01577	0.01761	0.01580	0.01480	0.01574	0.02214	0.01793	0.01630
β_2	0.01127	0.01123	0.01171	0.01061	0.01006	0.01127	0.01534	0.01266	0.01166
β_3	0.78666	0.78152	0.67763	0.62479	0.60392	0.78662	0.95081	0.81191	0.77010
β_4	-0.47132	-0.46992	-0.48975	-0.44423	-0.42170	-0.47131	-0.63960	-0.52621	-0.48340
β_5	-1.32410	-1.31870	-1.24442	-1.13786	-1.08974	-1.32405	-1.68676	-1.41595	-1.32408
β_6	-0.75584	-0.75594	-0.86971	-0.78142	-0.73330	-0.75583	-1.09894	-0.88944	-0.80596
β_7	-2.20748	-2.20104	-2.48191	-2.23117	-2.09511	-2.20743	-3.15262	-2.56252	-2.33208
β_8	-0.96060	-0.95756	-0.99906	-0.90459	-0.85667	-0.96058	-1.30317	-1.07358	-0.98838
β_9	-0.24569	-0.24476	-0.21884	-0.20160	-0.19465	-0.24568	-0.30509	-0.25879	-0.24367

rate BMPC protocol takes about 3 minutes, and using Accurate BMPC protocol requires 5 minutes.

Figures 3 and 4 illustrate the CPU time of our protocols which are implemented based on algorithm 2 (BMP and CMPC). As shown in Figure 3, both of these protocols have a better performance than OLR when the number of records increases and the number of features is fixed. Also, BMPC has a considerably better CPU time in comparison with CMPC and OLR. However, as is shown in Figure 4, increasing the number of features has slightly different results. Raising the number of features decreases the efficiency of all three protocols. CMPC receives the highest impact from rising the number of features, but BMPC still has an acceptable efficiency level. For example, BMPC can train a model with 7000 records and 2500 features in less than one minute, which is 7 seconds higher than OLR, and four times better than CMPC. Therefore, we can conclude that CMPC is not the right choice when we have a dataset with a considerable number of features.

Besides, to measure the efficiency in terms of memory consumption, a python module called "memory_profiler" [32] has been utilized. Table 1 indicates memory consumption for different introduced protocols and ordinary logistic regression. All implemented protocols in both security settings consume remarkably less memory during the training process. As is describes in the table 1, to train a logistic regression model over a dataset with 50000 records and 200 features, the OLR protocol requires about 30 GB of memory. However, the BMPC and Accurate BMPC about 8 GB of memory to train this model. Also, the CMPC and Accurate CMPC protocols have a better consumption rate than BMPC, and less than 5 GB of memory is needed to train such a model.

5.3 Accuracy

One of the main subjects that we considered in examining our protocols is whether they are accurate. We measured the accuracy based on the estimated model parameters' precision during the training phase over the Low Birth Weight Study dataset. To do this, we compared the obtained vector of coefficient β from our protocols with the ones estimated using the OLR protocols. As Table 2 presents, the Accurate BMPC and Accurate CMPC protocols' model parameters are almost the same as the model parameters estimated using OLR protocols. Moreover, the model parameters estimated from protocols based on the algorithm 2 (BMPC and CMPC), which employ various approximations, have an acceptable level of accuracy compared to the model parameters estimated using OLR protocols.

Since we introduced several approximation schemes in BMPC and CMPC protocols (e.g., fixed-point Hessian matrix, the least-squares approximation of the Sigmoid function, and matrix inversion), and to have a better understanding of the accuracy level of these protocols, we compare the prediction accuracy achieved by these protocols with that obtained from OLR. To do this, we calculated the percentage (%) of the correct predictions of the models produced on four different datasets (25% of training samples were assigned to

Table 3: Accuracy comparison result for real-word datasets with different settings

				CMPC		BMPC		OLR	
Dataset	Records Num	Feature Num	g(x) degree	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
PIMA	768	9	3	71.87%	0.740	71.87%	0.740	71.87%	0.740
			5	71.87%	0.741	71.87%	0.740	71.87%	0.740
			7	71.87%	0.741	71.87%	0.741	71.87%	0.741
			No approx	-	-	-	-	71.87%	0.741
PCS	379	10	3	81.05%	0.842	81.05%	0.846	80%	0.846
			5	81.05%	0.845	81.05%	0.847	80%	0.847
			7	81.05%	0.847	81.05%	0.848	81.05%	0.848
			No approx	-	-	-	-	81.05%	0.848
LBW	189	10	3	64.58%	0.519	64.58%	0.519	64.58%	0.519
			5	64.58%	0.519	64.58%	0.519	64.58%	0.519
			7	62.5%	0.519	62.5%	0.517	64.58%	0.519
			No approx	-	-	-	-	62.5%	0.523
UIS	575	9	3	73.61%	0.651	73.61%	0.651	73.61%	0.651
			5	72.91%	0.652	72.91%	0.652	72.91%	0.652
			7	72.91%	0.655	73.61%	0.651	72.91%	0.655
			No approx	-	-	-	-	72.22%	0.656

the test set) in different settings based on the degree of Sigmoid function approximation. All the accuracy measurement results are summarized in Table 3. This table presents the average prediction accuracy percentage when threshold = 0.5 and the AUC (Area Under the Receiver Operating Characteristic Curve), which estimates a binary classifier's quality. It is clear from the table data that the approximations used in BMPC and CMPC protocols do not significantly affect the estimated model's accuracy. In other words, although based on the information provided in Table 2, varying the Sigmoid approximation degrees affect the exactness of model parameters learned during the training phase, these differences do not considerably impact the classification accuracy over the chosen datasets.

5.4 Communication cost

The efficiency of multi-party computation protocols in terms of communication costs is considered a fundamental concern. Since in our implementation, we simulate each computation party as a thread in a multi-threading setting, we theoretically compute the communication cost in this part. To do this, first, we compute the primary operations' communication cost, such as addition and multiplication, in both honest majority and dishonest majority settings. Secondly, we compute the total communication cost for the whole protocol based on the number of primary operations contained.

The addition operation in both security settings does not require any communication between the parties. However, multiplication requires multiple communication rounds. In the honest majority setting and with three computation parties, Bogdanov [33] explained that each time performing the MPC-based multiplication requires exchanging 15 messages between the computation parties. If we consider each message with the size of 32 bits, the communication cost for one-time multiplication will be 420 bits. Besides, one time executing our logistic regression protocol (CMPC) requires performing between 100 to 300 times multiplication protocol (based on the degree of least-squares Sigmoid approximation and choosing a good start value for the inversion operation). Therefore, to compute the logistic regression model in an honest majority setting with our protocol, at least 42 Kb data will be exchanged.

In the dishonest majority setting and using the Beaver multiplication approach, communication costs are lower than the honest majority setting. The multiplication procedure in this method is split into the offline and online phases. During the offline phase, multiplication triples will be generated and distributed before the computation parties' inputs be associated. Therefore, the communication cost of this phase can be safely ignored. During the online phase and in the two-party setting, each computation party sends only two messages to the other party to perform the multiplication. Accordingly, if we consider each message with the size of 32 bits, therefore 128-bits data require to be exchanged for performing one-time multiplication in this setting, which is 3.75 times less than the honest majority setting. For the whole logistic regression protocol, 12.5 Kb of data will be transferred.

6 Conclusions

There is an increasing interest in applying machine learning algorithms to sensitive data, such as medical data. In this paper, we described novel protocols for implementing secure and private logistic regression training among distributed parties using multi-party computation protocols. We evaluated the performance of our protocols through experiments on real-world

and synthetic datasets. With the latter, we showed that our solutions scale well when apply to a dataset with a very large number of records and features. Our experiments also showed that our protocol achieves high accuracy while maintaining a reasonable level of efficiency. In the future, we will extend our protocols to support secure and efficient multi-class logistic regression.

Declarations

Abbreviations

LR, logistic regression; MPC, multi-party computation; GLORE, grid binary logistic regression;SMAC-GLORE, Secure Multi-pArty Computation Grid LOgistic REgression; BMPC, beaver triple-based multi-party computation; CMPC, classical multi-party computation; OLR, ordinary logistic regression; PIMA, pima indians diabetes dataset; LBW, low birth weight study; PCS, prostate cancerstudy; UIS, umaru impact study datasets

Availability of data and materials

github.com/alirezaghavamipour/pplr_ss

Authors' contributions

The author ARG contributed the majority of the writing and conducted major parts of the study. FT wrote a part of this paper, provided detailed edits and critical suggestions. XJ provided the motivation for this work and helpful comments. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Funding

Not applicable.

Acknowledgment

We thank Aida Plocco and Sytze Tempel for their contribution to the idea of this study.

Author details

[1]University of Groningen, Nijenborgh 9, Groningen, Netherlands.
[2]UTHealth School of Biomedical Informatics, The University of Texas, Houston, US.

References

1. Hosmer Jr, D.W., Lemeshow, S., Sturdivant, R.X.: Applied logistic regression. (2013). John Wiley & Sons
2. Boxwala, A.A., Kim, J., Grillo, J.M., Ohno-Machado, L.: Using statistical and machine learning to help institutions detect suspicious access to electronic health records. Journal of the American Medical Informatics Association **18**(4), 498–505 (2011)
3. Riley, R.D., Ensor, J., Snell, K.I., Harrell, F.E., Martin, G.P., Reitsma, J.B., Moons, K.G., Collins, G., van Smeden, M.: Calculating the sample size required for developing a clinical prediction model. Bmj **368** (2020)
4. Jagadeesh, K.A., Wu, D.J., Birgmeier, J.A., Boneh, D., Bejerano, G.: Deriving genomic diagnoses without revealing patient genomes. Science **357**(6352), 692–695 (2017)
5. Shi, H., Jiang, C., Dai, W., Jiang, X., Tang, Y., Ohno-Machado, L., Wang, S.: Secure multi-party computation grid logistic regression (smac-glore). BMC medical informatics and decision making **16**(3), 89 (2016)
6. Xie, W., Wang, Y., Boker, S.M., Brown, D.E.: Privlogit: Efficient privacy-preserving logistic regression by tailoring numerical optimizers. arXiv preprint arXiv:1611.01170 (2016)
7. De Cock, M., Dowsley, R., Horst, C., Katti, R., Nascimento, A.C., Poon, W.-S., Truex, S.: Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. IEEE Transactions on Dependable and Secure Computing **16**(2), 217–230 (2017)
8. Beaver, D.: Commodity-based cryptography. In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, pp. 446–455 (1997)
9. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38 (2017). IEEE
10. Gentry, C., Boneh, D.: A Fully Homomorphic Encryption Scheme vol. 20. Stanford university Stanford
11. Yoo, J.S., Hwang, J.H., Song, B.K., Yoon, J.W.: A bitwise logistic regression using binary approximation and real number division in homomorphic encryption scheme. In: International Conference on Information Security Practice and Experience, pp. 20–40 (2019). Springer
12. MLD, R., Fienberg, S., Nardi, Y.: Secure multiparty linear and logistic regression based on homomorphic encryption. cs.cmu.edu. Query date: 2020-06-24 08:59:23
13. Carpov, S., Gama, N., Georgieva, M., Troncoso-Pastoriza, J.R.: Privacy-preserving semi-parallel logistic regression training with fully homomorphic encryption. BMC Medical Genomics **13**(7), 1–10 (2020)
14. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics **6**(2), 19 (2018)
15. Han, K., Hong, S., Cheon, J.H., Park, D.: Logistic regression on homomorphic encrypted data at scale. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 9466–9471 (2019)
16. Han, K., Hong, S., Cheon, J., Park, D.: Efficient logistic regression on large encrypted data. IACR Cryptol. ePrint Arch. (2018). Query date: 2020-06-24 08:59:23
17. Djonatan, P.: Privacy-preserving Analytics: Secure Logistic Regression, (2019). Query date: 2020-06-24 08:59:23. https://dr.ntu.edu.sg/handle/10356/77126
18. Du, W., Li, A., Li, Q.: Privacy-preserving multiparty learning for logistic regression. In: International Conference on Security and Privacy in Communication Systems, pp. 549–568 (2018). Springer
19. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Advances in Neural Information Processing Systems, pp. 289–296 (2009)
20. El Emam, K., Samet, S., Arbuckle, L., Tamblyn, R., Earle, C., Kantarcioglu, M.: A secure distributed logistic regression protocol for the detection of rare adverse drug events. Journal of the American Medical Informatics Association **20**(3), 453–461 (2013)
21. Kim, M., Lee, J., Ohno-Machado, L., Jiang, X.: Secure and differentially private logistic regression for horizontally distributed data. IEEE Transactions on Information Forensics and Security **15**, 695–710 (2019)
22. Bogdanov, D., Niitsoo, M., Toft, T., Willemson, J.: High-performance secure multi-party computation for data mining applications. International Journal of Information Security **11**(6), 403–418 (2012)
23. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference, pp. 420–432 (1991). Springer
24. Nardi, Y., Fienberg, S.E., Hall, R.J.: Achieving both valid and secure logistic regression analysis on aggregated data from different private sources. Journal of Privacy and Confidentiality **4**(1) (2012)
25. Agresti, A.: Categorical data analysis **482** (2003)
26. Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR medical informatics **6**(2), 19 (2018)
27. Böhning, D.: The lower bound method in probit regression. Computational statistics & data analysis **30**(1), 13–17 (1999)
28. Dua, D., Graff, C.: UCI Machine Learning Repository (2017). http://archive.ics.uci.edu/ml
29. lbw: Low Birth Weight study data (2019). https://rdrr.io/rforge/LogisticDx/man/lbw.html
30. pcs: Prostate Cancer Study data (2019). https://rdrr.io/rforge/LogisticDx/man/pcs.html

31. uis: UMARU IMPACT Study data (2019). https://rdrr.io/rforge/LogisticDx/man/uis.html
32. memory-profiler (2021). https://pypi.org/project/memory-profiler/
33. Bogdanov, D.: Sharemind: programmable secure computations with practical applications. PhD thesis, Tartu University (2013)