Privacy-Preserving Logistic Regression as a Cloud Service based on Residue Number System

Jorge Mario Cortés-Mendoza[1][0000-0001-7209-8324], Andrei Tchernykh[1,2,4 [0000-0001-5029-5212]], Mikhail Babenko[3][0000-0001-7066-006], Luis Bernardo Pulido-Gaytán[2][0000-0002-7384-7670], Gleb Radchenko[1][0000-0002-7145-5630], Franck Leprevost[5][0000-0001-8808-2730], Xinheng Wang[6][0000-0001-8771-8901], and Arutyun Avetisyan[4][[0000-0002-0470-9944].

[1] South Ural State University, Prospekt Lenina 76, 454080, Chelyabinsk, Russia.
[2] CICESE Research Center, Carr. Tijuana-Ensenada 3918, 22860, Ensenada, BC, Mexico.
[3] North-Caucasus Federal University, Kulakova 2, 355029, Stavropol, Russia.
[4] Ivannikov Institute for System Programming, Solzhenitsyn 25, 109004, Moscow, Russia.
[5] University of Luxembourg, Ave. de l'Université 2, L-4365, Esch-sur-Alzette, Luxembourg.
[6] Xi'an Jiaotong-Liverpool University, Ren'ai Road 111, 215123, Suzhou, China.

kortesmendosak@susu.ru, chernykh@cicese.mx, mgbabenko@ncfu.ru, lpulido@cicese.edu.mx, gleb.radchenko@susu.ru, franck.leprevost@uni.lu, xinheng.wang@xjtlu.edu.cn, arut@ispras.ru

Abstract. The security of data storage, transmission, and processing is emerging as an important consideration in many data analytics techniques and technologies. For instance, in machine learning, the datasets could contain sensitive information that cannot be protected by traditional encryption approaches. Homomorphic encryption schemes and secure multi-party computation are considered as a solution for privacy protection. In this paper, we propose a homomorphic Logistic Regression based on Residue Number System (LR-RNS) that provides security, parallel processing, scalability, error detection, and correction. We verify it using six known datasets from medicine (diabetes, cancer, drugs, etc.) and genomics. We provide experimental analysis with 30 configurations for each dataset to compare the performance and quality of our solution with the state of the art algorithms. For a fair comparison, we use the same 5-fold cross-validation technique. The results show that LR-RNS demonstrates similar accuracy and performance of the classification algorithm at various thresholds settings but with the reduction of training time from 85.4% to 97.5%.

Keywords: Cloud Security, Homomorphic Encryption, Residue Number System, Logistic Regression.

1 Introduction

The cloud computing paradigm provides an easy way to store, retrieve, and process data as a part of its services. Machine Learning as a Service (MLaaS) has emerged as a flexible and scalable solution [1, 2, 3]. Unfortunately, security and privacy issues still

pose significant challenges, especially when data must be decrypted, for instance, for analytics.

Homomorphic Encryption (HE), Fully Homomorphic Encryption (FHE), Somewhat Homomorphic Encryption (SHE), and secure Multi-Party Computation (MPC) are ways to address vulnerabilities of data processing. These cryptosystems allow applying certain mathematical operations directly to the ciphertext and safely delegate the processing of data to an untrusted remote party, it guarantees that the remote party will learn neither the input nor the output of the computation [4].

In the last decade, there is considerable interest in using the Residue Number System (RNS) as a variant of FHE [5]. It is a widely known and studied number theory system. It codes the original number as a tuple of residues with respect to a moduli set that can be processed in parallel. This coding technique is one of the core optimization techniques used in several implementations of HE schemes. The advantages of RNS include security, parallel processing, scalability, error detection, and correction.

The paper focuses on developing RNS Logistic Regression (LR-RNS) for the processing of confidential information in cloud computing. The goal is to enrich the paradigm of Machine Learning as a Service. Our contribution is multifold:

- We propose a logistic regression algorithm with a homomorphic encryption scheme based on a residue number system.
- Training, testing, and prediction process are performed with ciphertexts.
- We conduct comprehensive simulation with six known datasets of different domains in medicine (diabetes, cancer, drugs, etc.) and genomics.
- We show that LR-RNS has similar accuracy and classification performance compared with the state of the art algorithms, with considerable training time decrease.

The rest of the paper is structured as follows. Section 2 discusses related works to solve our problem. Section 3 presents information about cloud security, homomorphic encryption, and residue number system. Section 4 describes the characteristics of the logistic regression in RNS. Section 5 presents the experimental results. Finally, we conclude and discuss future work in Section 6.

2 Logistic Regression and Gradient Descent

Logistic Regression (LR) is a simple and powerful strategy to solve problems in different domains: detection of prostate cancer, diabetes, myocardial infarction, infant mortality rates, cardiac problems, treatment for drug abuse, genomic, fraudulent transactions detection, etc. [10 - 15]. It is a statistical method for analyzing information with independent variables. It determines a binary (or dichotomous) outcome (success/failure, yes/no, etc.) using logistic functions to predict a dependence on the data. A dataset with dimension d defined by $X^{(i)} \in \mathbb{R}^d$ and their corresponding labels $Y^{(i)} \in \{0,1\}$ for $i = 1, 2, \dots, n$ are used to model a binary dependent variable. The inference of logistic regression is considered within hypotheses $h_\theta\left(X^{(i)}\right) = g\left(\theta^T X^{(i)}\right)$, where the sigmoid function is defined as $g(z) = \frac{1}{1+e^{-z}}$ and $\theta^T X^{(i)} = \theta_0 + \theta_1 {X_1}^{(i)} + \theta_2 {X_2}^{(i)} + \cdots + \theta_d {X_d}^{(i)}$, for $\theta^T = [\theta_0, \theta_1, \dots, \theta_d]^T$ and $X^{(i)} = [1, {X_1}^{(i)}, {X_2}^{(i)}, \dots, {X_d}^{(i)}]^T$.

Logistic regression uses a likelihood function to make inference on parameter θ. Simplifying the likelihood function for the whole data by log yields

$J(\theta) = \sum_{i=1}^{n} Y^{(i)} \log\left(h_\theta\left(X^{(i)}\right)\right) + \left(1 - Y^{(i)}\right) \log\left(1 - h_\theta\left(X^{(i)}\right)\right)$.

Techniques to minimize $J(\theta)$ vary, however, the gradient descent algorithm is a common option.

Gradient Descent (GD) is an optimization algorithm to minimize the objective function $J(\theta)$. At each iteration, it updates the parameters θ in the opposite direction of the gradient of the function. The learning rate α determines the dimension of the steps to reach a (local) minimum. The direction of the slope, created by the objective function, guides the search downhill until to reach a valley.

Batch Gradient Descent (BGD) is the most common version of GD. It updates the values of θ considering the entire training dataset. BGD guarantees to converge to the global minimum for convex error surfaces and local minimum for non-convex surfaces. But it has a slow time of convergence. Big datasets can be intractable due to memory limitations and low access speed.

Stochastic Gradient Descent (SGD) performs an update of θ for each training example. Advantages of SGD include reduced convergence time, exploration of new valleys (potentially with better local minima), and online learning.

In order to estimate the parameter θ, GD with Fixed Hessian Newton method (GD-FHN) applies the Newton-Raphson method to solve the equation numerically which iteratively determines the zeros of a function. The Hessian matrix is the second partial derivative of $J(\theta)$, but its evaluation and inverse are quite expensive. So, GD-FHN simplifies the process by approximating the Hessian matrix. Momentum is a method to accelerate the direction and reduce the oscillations of SGD. It defines a fraction γ and uses it to update θ. Nesterov Accelerated Gradient (NAG) takes advantage of the momentum term to improve the performance of GD. It provides an approximation of the next position of the parameters with partial updates of θ.

The training phase of logistic regressions focuses on finding values θ^* that minimizes the cost function $J(\theta)$. θ^* values are used to estimate the binary classification of new data. For example, for a given data $X = [1, X_1, X_2, \ldots, X_d] \in \mathbb{R}^{d+1}$, it is possible to guess its binary value $Y \in \{0,1\}$ by setting:

$$Y = \begin{cases} 1 \; if \; h_{\theta^*}(\mathrm{X}) \geq \tau \\ 0 \; if \; h_{\theta^*}(\mathrm{X}) < \tau \end{cases} \quad (1)$$

where τ defines a variable threshold in $0 < \tau < 1$, typically with value equal to 0.5.

3 Related Work

Homomorphic Encryption (HE) is an active research field and has a long list of approaches and improvements [6 - 9]. Solutions focus on algorithms, arithmetic operations, approximation functions, applications for data analytics, Machine Learning (ML) [10 - 15], Internet of Things (IoT) [22], etc.

Aono et al. [10] propose a homomorphism-aware LR system where the training and predicting data are protected under encryption. The authors study the quality of the

additive homomorphic encryption schemes with the public key (Paillier), Learning With Errors (LWE), and ring-LWE (see below).

Bonte et al. [11] develop a privacy-preserving logistic regression using SHE. The central server can combine the data of several users to train the binary classification model without learning anything about the underlying information.

Kim et al. [12] present a method to train logistic regression with a SHE. The authors use NAG method to increase the speed of convergence and a packing method to reduce the storage of encryption data.

Cheon et al. [13] define a variant of HE scheme for Arithmetics of Approximate Numbers (HEAAN) based on the RNS representation of polynomials in a software library that implements homomorphic encryption and supports fixed-point arithmetic. The algorithm uses RNS decomposition of cyclotomic polynomials and Number Theoretic Transformation (NTT) conversion.

Cheon et al. [14] propose a variation of GD for logistic regression. Ensemble gradient descend runs several standard GDs on a partial dataset and then takes an average on resulting solutions (from each of the partial datasets). The algorithm reduces the number of iterations to train the model, hence, improving the execution time of logistic regression. However, the errors from approximate computations may disrupt the convergence of the ensemble algorithm in an encrypted state.

Yoo et al. [15] develop an LR for HE using binary approximation. It can represent reals numbers, perform subtraction, division, and exponential functions based on the encrypted bitwise operation.

Table 1 summarizes the main characteristics of the approaches.

Table 1. Main characteristics of HE schemes for logistic regression.

Encryption	Evaluation function	Gradient descent	Metrics (Section 5.1)	Library	Datasets	Ref.
Paillier, LWE, Ring-LWE	Taylor series	BGD	F-score, AUC	-	Pima, SPECTF	[10]
Ring-LWE	Taylor series	GD-FHN	ROC, accuracy	NFLlib	iDASH, financial data	[11]
Ring-LWE	Polynomials of degree 3, 5 and 7	NAG	AUC, accuracy	HEAAN	iDASH, lbw, mi, nhanes3, pcs, uis	[12]
Ring-LWE, RNS	A polynomial of degree 7	NAG	AUC, accuracy	HEAAN	Lbw, uis	[13]
Ring-LWE	A polynomial of degree 5	NAG	AUC	HEAAN	MNIST, credit	[14]
	Logistic function	BGD	AUC	-	NIDDK	[15]

Paillier encryption scheme is a probabilistic asymmetric cryptography algorithm. It consists of a public key $pk = n$ used to encrypt plaintext in the interval $\{0, \dots, n-1\}$. The additive homomorphic encryption defines $PaiEnc_{pk}(m, r) = g^m r^n mod\ n^2$ to encrypt a message m, where $n = p * q$ for primes numbers p and q, a randomly selected $r \in \{0, \dots, n-1\}$, and integer $g \in \mathbb{Z}^*_{n^2}$. The encryption function $PaiEnc_{pk}(m, r)$ has the additively homomorphic property $PaiEnc_{pk}(m_1, r_1) * PaiEnc_{pk}(m_2, r_2) = PaiEnc_{pk}(m_1 + m_2, r_1 r_2)$.

LWE encryption scheme provides a public key $pk = (A, P, (p, l), (n_{lwe}, s, q))$ in the interval $\mathbb{Z}_p = (-p/2, p/2]$ to represent plaintext in a vector $\mathbb{Z}_p^l$, where p and l are security parameters, and A with P define a matrix concatenation of public matrices $A \in \mathbb{Z}_p^{n_{lwe} \times n_{lwe}}$. For a plaintext $m \in \mathbb{Z}_p^{1 \times l}$, the encryption message m can be generated by: $LweEnc_{pk}(m) = e_1[A|P] + p[e_2|e_3] + [0_{n_{lwe}}|m] \in \mathbb{Z}_q^{1 \times (n_{lwe}+l)}$, in which $e_1 \in \mathbb{Z}_q^{1 \times n_{lwe}}$, $e_2 \in \mathbb{Z}_q^{1 \times n_{lwe}}$, $e_3 \in \mathbb{Z}_q^{1 \times l}$ are Gaussian noise vectors of deviation s. The additive encryption produces ciphertext $m_1 + m_2$ by $LweEnc_{pk}(m_1) + LweEnc_{pk}(m_2)$.

Ring LWE encryption scheme defines a public key $pk = (a, p) \in R_p^2$ in the interval $\mathbb{Z}_p = (-p/2, p/2] \cap \mathbb{Z}$ with a ring $R = \mathbb{Z}[x]/f(x)$ where $f(x) = x^{n_{rlwe}} + 1$ and quotient rings $R_q = R/q, R_p = R/p$. The additive homomorphic encryption generates a ciphertext with the message m by: $RlweEnc_{pk}(m) = (e_1 a + e_2, e_1 p + e_3 + \lfloor p/q \rfloor m)$, where $e_1, e_2, e_3 \in R_{(0,s)}$ are noises and $R_{(0,s)}$ stand for polynomials in R with small Gaussian coefficients of mean 0 and deviation s. The addition of the two ciphertexts m_1 and m_2 can be done as $RlweEnc_{pk}(m_1) + RlweEnc_{pk}(m_2)$.

4 Security Methods

Cloud computing provides data protection from theft, leakage, deletion, integrity, etc. at levels of firewalls, penetration testing, obfuscation, tokenization, Virtual Private Networks (VPN), etc. However, the use of third-party services can bring several cybersecurity risks. Using conventional data encryption does not avoid the problem. At some point, the data must be decrypted for processing, for instance, for statistical analysis or training a logistic regression model. At this moment, the vulnerability of the information is high. A possible solution is to use encryption schemes that allow performing operations over the ciphertext.

4.1 Homomorphic Encryption

HE allows performing operations on ciphertexts based on publicly available information without having access to any secret key.

An additively homomorphic encryption scheme generates ciphertexts c_1 and c_2 with the encrypted content of the messages m_1 and m_2, respectively. Then, decryption of the ciphertext $\mathrm{c}_+ = \mathrm{c}_1 + \mathrm{c}_2$ produces $\mathrm{m}_1 + \mathrm{m}_2$. Similarly, a multiplicatively homomorphic encryption scheme produces a ciphertext $\mathrm{c}_\times$ that decrypts $\mathrm{m}_1 \times \mathrm{m}_2$.

It includes multiple types of encryption schemes and has been developed using different approaches: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Full Homomorphic Encryption (FHE). PHE supports only one type of operation, for example, a scheme supports homomorphic addition but not multiplication. PHE is generally more efficient than SHE and FHE. SHE supports additions and multiplications but can perform only a limited number of operations before the error grows too much to maintain the correctness of the evaluation. FHE allows an increased number of operations due to bootstrapping. Approximate computation im-

proves the efficiency of the FHE cryptosystem. In the last decade, there has been considerable interest in using Residue Number System (RNS) in FHE schemes. Different HE cryptosystems are applied for cloud computing: RSA, Paillier, El Gamal, Goldwasser-Micali, Boneh-GohNissim, and Gentry [20].

4.2 Residue Number System

Residue Number System (RNS) is a variation of finite ring isomorphism widely known and studied. The RNS represents original numbers as residues over the moduli set. The advantages of this representation include inherently carry-free operations, smaller numbers that encode original numbers, and no-positional system with independent arithmetic units. The use of RNS has gained considerable interest in HE schemes due to some characteristics of the system: security, parallel processing, scalability, error detection, and correction [16, 17].

A moduli set of pairwise co-prime numbers $\{p_1, p_2, \dots, p_n\}$ defines the representation of the values and the range $P = \prod_1^n p_i$. An integer number X, where $X \in [0, P-1)$, is defined in RNS as a tuple $(x_1, x_2, \dots, x_n)$ where x_i represents the remainder of the division of X by p_i, defined by $x_i = |X|_{p_i}$.

The RNS system also allows performing arithmetic operations with several properties. For instance, given X and Y integer numbers, and their representation in RNS by the tuples $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ then:

$$X \otimes Y = (x_1, x_2, \dots, x_n) \otimes (y_1, y_2, \dots, y_n) = \\ (|x_1 \otimes y_1|_{p_1}, |x_2 \otimes y_2|_{p_2}, \dots, |x_n \otimes y_n|_{p_n}) = (z_1, z_2, \dots, z_n) = Z \quad (2)$$

where $\otimes$ denotes one operation: addition, multiplication, and subtraction; with $z_i = |Z|_{p_i}$, for all $i = 1, n$.

Eq. 2 shows that the RNS can be defined as a variant of HE [22]. We can obtain secure data processing and storage since the representation of numbers in RNS can be seen as coding and secret sharing scheme. For every n-tuple, the corresponding integer $X \in [0, P-1)$ can be recovered by means of the Chinese Remainder Theorem (CRT) $X = (\sum_{i=1}^n x_i P_i b_i) mod\ P, \forall i = 1, \dots, n$, where $P_i = P/p_i$ and b_i is the multiplicative inverse of $P_i\ mod\ p_i$.

Additionally, in order to avoid using real $X_R \in R$ numbers, we use the scaling factor method to represent real values in the integer domain, where $X = \lfloor 2^{powerInt} X_R \rfloor$. In general, operations that require the magnitude of a number are high complexity in the RNS, for instance, magnitude comparison, sign and overflow detection, division, etc. More efficient algorithms to compute division [18] and comparison of numbers in RNS were proposed in [19].

4.3 Logistic function approximation

To implement a logistic regression in RNS, we approximate the standard logistic function $g(z)$ and $J(\theta)$ (see Section 2) using first-degree polynomials. On one hand, a re-

striction on the degree of a polynomial arises due to restrictions on the number of multiplication operations that can be performed with encrypted text using homomorphic encryption. On the other hand, there is a great computational complexity of the multiplication operation in comparison with the addition in homomorphic encryption.

As a simplest initial approximation, we use the function defined by three lines $y_1(z) = 0$ if $z < -2$, $y_2(z) = \frac{1}{4}(2 + z)$ if $-2 \leq z \leq 2$, and $y_3(z) = 1$ if $z > 2$ (see Figure 1). $y_2(z)$ is the tangent to the reference sigmoid function in point $z = 0$. The maximum approximation error is $\frac{1}{1+e^2} \approx 0.1192$. Since the sigmoid function has the following property of $g(-z) = 1 - g(z)$, its calculation for a negative argument is equal to the calculation for a positive argument.

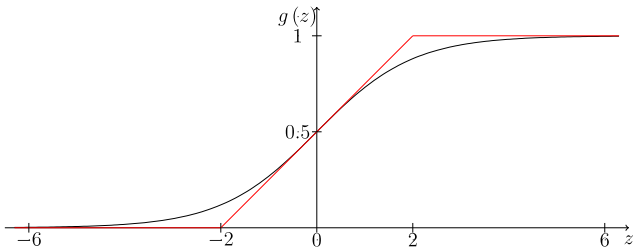

Fig. 1. Approximation function of $g(z)$

4.4 Homomorphic Logistic Regression with Residue Number System

We implemented the Batch Gradient Descendent algorithm with Residual Number System (LR-RNS) to provide security in the training and prediction phases of logistic regression. Both processes can be computed in a third-party infrastructure without worrying about the data leaking. At each iteration of the algorithm, all records in the training set are used to update the values of θ. After processing the training set, all theta values are updated at the same time.

Parallelism is one of the inherent advantages of RNS. The main loop of the algorithm can be done in parallel to reduce the execution time. Assuming that the number of resources is equal to the number of moduli, each resource performs the operation of each moduli for all the elements in the training set. It reduces the execution time of the loop by the number of elements in the moduli set. At the end of the execution, the algorithm returns θ^* to estimate the prediction values of new elements.

Before updating the values of θ in each iteration of the algorithm, a rescale function is used to eliminate the scaling factors. After each multiplication, a scaling factor is accumulated in the result of the operation. We use CRT to partially decode θ with an accumulated scaling factor, then adjust the value and encrypt the information again. The only risk of the data leaking is partial computing of θ.

5 Experimental results

In this section, we describe the evaluation method, configuration setup, and experimental results. The LR and LR-RNS algorithms were implemented using jdk 1.8.0_221 64-bit and Metrics library (in R).

5.1 Evaluation method

The efficiency of a classifier is defined by the number of correct and incorrect classification forms of each class. A Confusion Matrix (CM) displays the difference between the true and predicted classes for a set of examples. Accuracy (A) expresses the systematic error to estimate a value. Precision (P) defines the closeness of the measurements between classes. Recall (R) or sensitivity measures the proportion of positives values that are correctly identified. Specificity (S) measures the proportion of negative values (see Eq. 3).

$$A = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}\ ; P = \frac{T_p}{T_p + F_p};\quad R = \frac{T_p}{T_p + F_n}\ ; S = \frac{T_n}{T_n + F_p} \tag{3}$$

T_p defines the number of elements classified correctly as positive, T_n - correctly classified as negative, F_p incorrectly classified as positive, and F_n - incorrectly classified as negative. C_n presents the number of truly negative, C_p shows the truly positive examples, R_n defines the number of predicted negative, and R_p is the predicted positive examples; with a total of N elements.

The measures $A, P, R,$ and S depend on the threshold τ (see Eq. 1). It can increase or decrease the classification efficiency. The Receiver Operating Characteristic (ROC) allows plotting multiples values of R and S with respect to different values of τ. ROC curve shows the ability of a binary classifier to discriminate when the threshold is varied. It can be used as a tool to compare and select models. The Area Under the ROC Curve (AUC) is a way to estimate the efficiency of models with different thresholds. AUC is an indicator of the performance comparison of two classifiers. ROC curve, AUC, and A provide good guidance in order to evaluate the proposed algorithm.

5.2 Datasets

We consider six datasets widely used in the literature [10, 12, 13, 21] (see Table 2). They define a series of continuous input variables and two output classes. To normalize the values of the features in the range of [0, 1], we use known min-max normalization, the simplest method using the formula $x'=\frac{x-\min(x)}{\max(x)-\min(x)}$, where x is an original value, and x' is the normalized value.

Low Birth Weight (Lbw) dataset consists of information about births to women in an obstetrics clinic. Myocardial Infarction (Mi) is a heart disease dataset. National Health and Nutrition Examination Survey (Nhanes3) includes a database of human exposomes and phenomes. Pima is the Indian's diabetes dataset. Prostate Cancer Study

(Pcs) dataset of patients with and without cancer of prostate. Umaru Impact Study (Uis) dataset stores information about resident treatment for drug abuse.

Table 2. Datasets characteristics and size of sets.

Dataset	Name	N	Features	N-Training	N-Testing
Lbw	Low Birth Weight study	189	9	151	38
Mi	Myocardial Infarction	1,253	9	1,002	251
Nhanes3	National Health and Nutrition Examination	15,649	15	12,519	3,130
Pima	Indian's diabetes	768	8	614	154
Pcs	Prostate Cancer Study	379	9	303	76
Uis	Umaru Impact Study	575	8	460	115

We use the 5-fold Cross-Validation (CV) technique to compare the performance of our algorithms and consider other similar results in the literature. 5-fold technique divides the dataset randomly in five subsets, four of them are used to train the prediction model, and the last set is used to validate the model.

5.3 Configuration setup.

We perform a series of experiments to find the best setup for LR and LR-RNS. The accuracy is calculated with a threshold of 0.5 and a scaling factor of 32 bits to represent real values with integer values. Our moduli set contains seven pair-wise relatively primes p_i: 18446744073709551629, 18446744073709551653, 18446744073709551667, 18446744073709551697, 18446744073709551709, 18446744073709551757, and 18446744073709551923.

Tables 3 and 4 present AUC and accuracy for LR and LR-RNS, respectively, for Lbw dataset. Each value represents the average of 30 execution with different initial values for θ. The idea is to define a set of values to analyze the behavior of the LR and LR-RNS algorithms. We use 6 values for α = {0.006, 0.001, 0.0006, 0.0001, 0.000006, and 0.00001}. The numbers of iteration of the algorithm are 100, 250, 500, 750, and 1,000.

Table 3. Average AUC and accuracy for several LR configurations with Lbw.

	AUC					**Accuracy**				
Iter / α	100	250	500	750	1,000	100	250	500	750	1,000
0.006	0.7612	**0.7693**	0.7642	0.7660	0.7628	0.7447	0.7412	0.7272	0.7105	0.7105
0.001	0.6559	0.7413	0.7585	0.7628	0.7667	0.6886	0.7149	0.7368	0.7474	**0.7544**
0.0006	0.6017	0.7056	0.7504	0.7584	0.7610	0.6860	0.6982	0.7167	0.7342	0.7439
0.0001	0.5188	0.5407	0.5863	0.6232	0.6558	0.6053	0.6614	0.6816	0.6877	0.6886
0.00006	0.5114	0.5245	0.5494	0.5784	0.6017	0.5702	0.6368	0.6667	0.6816	0.6842
0.00001	0.4985	0.5014	0.5087	0.5126	0.5189	0.5246	0.5412	0.5579	0.5895	0.6044

The worst configuration for AUC for both metrics and both algorithms is α = 0.0001 and 100 iterations. The best configuration for AUC is α = 0.006 with 250 iterations for

LR and α = 0.001 and 1,000 iterations for LR-RNS. The best accuracy is provided by the configuration α = 0.001 with 1,000 iterations for LR and α = 0.001 with 750 iterations for LR-RNS.

Table 4. Average AUC and accuracy for several LR-RNS configurations with Lbw.

	AUC					**Accuracy**				
Iter / α	100	250	500	750	1,000	100	250	500	750	1,000
0.006	0.7557	0.7583	0.7527	0.7476	0.7468	0.7298	0.7360	0.7368	0.7368	0.7368
0.001	0.6612	0.7420	0.7551	0.7564	**0.7585**	0.6904	0.7132	0.7307	**0.7377**	0.7298
0.0006	0.6058	0.7101	0.7504	0.7552	0.7557	0.6816	0.7009	0.7193	0.7272	0.7298
0.0001	0.5175	0.5417	0.5884	0.6278	0.6594	0.6000	0.6596	0.6807	0.6833	0.6904
0.00006	0.5120	0.5253	0.5530	0.5794	0.6045	0.5693	0.6360	0.6684	0.6807	0.6816
0.00001	0.4986	0.5012	0.5082	0.5122	0.5175	0.5237	0.5404	0.5570	0.5860	0.6009

5.4 Experimental analysis

Tables 5 and 6 show results for the datasets described in Table 2 and configurations described in Section 5.3. We also consider the results of Kim [12] and Aono [10] marked * and **, respectively, as references because a direct comparison is not fair due to different configurations and parameters.

Table 5. Average accuracy and AUC for datasets with 5-fold CV after 30 execution.

Name	Accuracy (%)			AUC		
	LR	LR-RNS	Other	LR	LR-RNS	Other
Lbw	74.12	72.98	69.19*	0.7693	**0.7585**	0.689*
Mi	88.45	88.66	91.04*	0.9486	0.9407	**0.958***
Nhanes3	81.40	85.41	79.22*	0.9027	**0.9016**	0.717*
Pcs	70.26	69.34	68.27*	0.7598	**0.7560**	0.740*
Pima	75.73	77.40	80.7 **	0.8575	0.8574	**0.8763****
Uis	74.55	74.78	74.44*	0.7338	**0.6343**	0.603*

LR-RNS provides similar results to LR, even with the use of the approximation function. Table 5 shows that for six datasets, the LR-RNS has similar average accuracy and AUC compared with other approaches. Table 6 shows the best solutions for datasets with 5-fold CV. It confirms that LR-RNS provides similar results to RL.

Table 6. Best accuracy and AUC for datasets with 5-fold CV.

Name	Accuracy (%)			AUC		
	LR	LR-RNS	Other	LR	LR-RNS	Other
Lbw	68.42	68.42	69.19*	0.8045	**0.8013**	0.689*
Mi	88.84	89.24	91.04*	0.9496	0.9496	**0.958***
Nhanes3	83.93	85.71	79.22*	90.37	**0.902**	0.717*
Pcs	75	75	68.27*	0.7788	**0.7772**	0.740*
Pima	76.03	78.76	80.7**	0.8581	0.8577	**0.8763****
Uis	74.78	74.78	74.44*	0.7478	**0.6343**	0.603*

Table 7 presents computing and updating times of theta per iteration for all datasets. Table 8 provides the learning time of the LR-RNS with 1,000 iterations. Training time decrease of LR-RNS is between 85-97%, 2.48%, and 14.58 % of time reported in the literature for five of the six datasets.

Table 7. Execution time per iteration of the algorithm (milliseconds).

Name	Processing theta	Updating theta
Lbw	6.6	1.1
Mi	29.4	1.1
Nhanes3	1,313.6	2.9
Pcs	16.5	1.0
Pima	33.6	0.9
Uis	21.7	1.2

Table 8. Leaning time for datasets with 5-fold CV and 1,000 iterations.

Name	Kim [12] (sec.)	LR-RNS (sec.)	Time decrease (%)
Lbw	3.3	0.082	97.52
Mi	3.6	0.525	85.42
Nhanes3	7.3	12.48	-70.95
Pcs	3.5	0.224	93.60
Pima	-	0.685	-
Uis	3.5	0.286	91.83

LR-RNS has worse performance in the Nhanes3 dataset increasing time about 70.95%. Reducing the number of iterations to the half, hence, reducing time twice, LR-RNS provides a solution of 0.8% lesser than the value reported in Table 5 for AUC.

6 Conclusion

The confidentiality of data is fundamental for cloud users. The cloud services should provide security of data storage and processing at any moment. In this paper, we propose a data confidentiality logistic regression algorithm for cloud service with homomorphic encryption based on a residue number system. We provide an experimental evaluation of its performance with several configurations and datasets of different domains in medicine (diabetes, cancer, drugs, etc.) and genomics. The training, testing, and prediction process are performed with ciphertexts. We show that LR-RNS has similar quality results in accuracy and AUC compared with the state of the art of homomorphic encryption LR algorithms but with a considerable time decrease of the training. However, further study is required to assess its actual efficiency and effectiveness in real systems. This will be the subject of future work on a real cloud environment.

References

1. Google, https://cloud.google.com/products/ai, last accessed 2020/03/13.
2. Microsoft, https://azure.microsoft.com/es-es/services/machine-learning, last accessed 2020/03/13.
3. Amazon, https://aws.amazon.com/machine-learning, last accessed 2020/03/13.
4. CSO, https://www.csoonline.com/article/3441477/enabling-public-but-secure-deep-learning.html, last accessed 2020/03/13.
5. Chervyakov, N., Babenko, M., Tchernykh, A., Kucherov, N., Miranda-López, V., Cortés-Mendoza, J.M.: AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. Futur. Gener. Comput. Syst. 92, 1080–1092 (2019). https://doi.org/10.1016/j.future.2017.09.061.

6. PALISADE, https://palisade-crypto.org/community, last accessed 2020/03/13.
7. Halevi, S., Shoup, V.: Algorithms in HElib. Presented at the (2014). https://doi.org/10.1007/978-3-662-44371-2_31.
8. HEANN, https://github.com/snucrypto/HEAAN, last accessed 2020/03/13.
9. SEAL, https://www.microsoft.com/en-us/research/project/microsoft-seal, last accessed 2020/03/13.
10. Aono, Y., Hayashi, T., Trieu Phong, L., Wang, L.: Scalable and Secure Logistic Regression via Homomorphic Encryption. In: Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy - CODASPY '16. pp. 142–144. ACM Press, New York, New York, USA (2016). https://doi.org/10.1145/2857705.2857731.
11. Bonte, C., Vercauteren, F.: Privacy-preserving logistic regression training. BMC Med. Genomics. 11, 86 (2018). https://doi.org/10.1186/s12920-018-0398-y.
12. Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC Med. Genomics. 11, 83 (2018).
13. Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A Full RNS Variant of Approximate Homomorphic Encryption. Presented at the (2019). https://doi.org/10.1007/978-3-030-10970-7_16.
14. Cheon, J.H., Kim, D., Kim, Y., Song, Y.: Ensemble Method for Privacy-Preserving Logistic Regression Based on Homomorphic Encryption. IEEE Access. 6, 46938–46948 (2018).
15. 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. Presented at the (2019). https://doi.org/10.1007/978-3-030-34339-2_2.
16. Tchernykh, A., Babenko, M., Chervyakov, N., Cortes-Mendoza, J.M., Kucherov, N., Miranda-Lopez, V., Deryabin, M., Dvoryaninova, I., Radchenko, G.: Towards Mitigating Uncertainty of Data Security Breaches and Collusion in Cloud Computing. In: 2017 28th International Workshop on Database and Expert Systems Applications (DEXA). pp. 137–141. IEEE (2017). https://doi.org/10.1109/DEXA.2017.44.
17. Tchernykh, A., Miranda-López, V., Babenko, M., Armenta-Cano, F., Radchenko, G., Drozdov, A.Y., Avetisyan, A.: Performance evaluation of secret sharing schemes with data recovery in secured and reliable heterogeneous multi-cloud storage. Cluster Comput. 22, 1173–1185 (2019). https://doi.org/10.1007/s10586-018-02896-9.
18. Babenko, M., Chervyakov, N., Tchernykh, A., Kucherov, N., Shabalina, M., Vashchenko, I., Radchenko, G., Murga, D.: Unfairness Correction in P2P Grids Based on Residue Number System of a Special Form. In: 2017 28th International Workshop on Database and Expert Systems Applications (DEXA). pp. 147–151. IEEE (2017).
19. Babenko, M., Tchernykh, A., Chervyakov, N., Kuchukov, V., Miranda-López, V., Rivera-Rodriguez, R., Du, Z., Talbi, E.-G.: Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption. Program. Comput. Softw. 45, 532–543 (2019).
20. Tchernykh, A., Babenko, M., Chervyakov, N., Miranda-López, V., Kuchukov, V., Cortés-Mendoza, J.M., Deryabin, M., Kucherov, N., Radchenko, G., Avetisyan, A.: AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage. Int. J. Approx. Reason. 102, 60–73 (2018). https://doi.org/10.1016/j.ijar.2018.07.010.
21. Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S.: Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the Annual Symposium on Computer Application in Medical Care. p. 261 (1988).
22. Tchernykh, A., Babenko, M., Chervyakov, N., Miranda-Lopez, V., Avetisyan, A., Drozdov, A.Y., Rivera-Rodriguez, R., Radchenko, G., Du, Z.: Scalable Data Storage Design for Non-Stationary IoT Environment with Adaptive Security and Reliability. IEEE Internet Things J. 1–1 (2020). https://doi.org/10.1109/JIOT.2020.2981276.