# Secure Logistic Regression Based on Homomorphic Encryption: Design and Evaluation

Miran Kim[1], Yongsoo Song[1,2], Shuang Wang[1], Yuhou Xia[3], and Xiaoqian Jiang[1]

[1] University of California, San Diego, USA
{mrkim, yongsoosong, shw070,x1jiang}@ucsd.edu
[2] Seoul National University, Republic of Korea
lucius05@snu.ac.kr
[3] Princeton University, Princeton, USA
yuhoux@math.princeton.edu

**Abstract.** Learning a model without accessing raw data has been an intriguing idea to the security and machine learning researchers for years. In an ideal setting, we want to encrypt sensitive data to store them on a commercial cloud and run certain analysis without ever decrypting the data to preserve the privacy. Homomorphic encryption technique is a promising candidate for secure data outsourcing but it is a very challenging task to support real-world machine learning tasks. Existing frameworks can only handle simplified cases with low-degree polynomials such as linear means classifier and linear discriminative analysis.
The goal of this study is to provide a practical support to the mainstream learning models (e.g. logistic regression). We adapted a novel homomorphic encryption scheme optimized for real numbers computation. We devised (1) the least squares approximation of the logistic function for accuracy and efficiency (i.e., reduce computation cost) and (2) new packing and parallelization techniques.
Using real-world datasets, we evaluated the performance of our model and demonstrated its feasibility in speed and memory consumption. For example, it took about 116 minutes to obtain the training model from homomorphically encrypted Edinburgh dataset. In addition, it gives fairly accurate predictions on the testing dataset. We present the first homomorphically encrypted logistic regression outsourcing model based on the critical observation that the precision loss of classification models is sufficiently small so that the decision plan stays still.

**Keywords.** Homomorphic encryption, approximate arithmetic, logistic regression, gradient descent

## 1 Introduction

Biomedical data are highly sensitive and often contain important personal information about individuals. In the United States, health care data sharing is protected by the Health Insurance Portability and Accountability Act [30], whereas biomedical research practitioners are covered under federal regulation governing the "Common Rule", a federal policy that protects people who volunteer for federally funded research studies [26]. These policies set high standards on the protection of biomedical data and violations will lead to financial penalties and lost reputation. On the other hand, cloud computing, which significantly simplifies information technology environments, is the trend for data management and analysis. According to a recent study by Microsoft, nearly a third of organizations work with four or more cloud vendors [20]. The privacy concern, therefore, becomes a major hurdle for medical institutions to outsource data and computation to the commercial cloud. It is imperative to develop advanced mechanisms to assure the confidentiality of data to support secure analysis in the cloud environment.

An intuitive solution is to train a model without accessing the data and only obtain the estimated model parameters in a global manner. Assuming summary statistics can be shared, this can be done in a joint manner and we have developed the "grid logistic regression" [33, 17, 32] to show the feasibility of estimating the global parameters from distributed sources (e.g. by only exchanging gradients and Hessian matrices). However, there are still

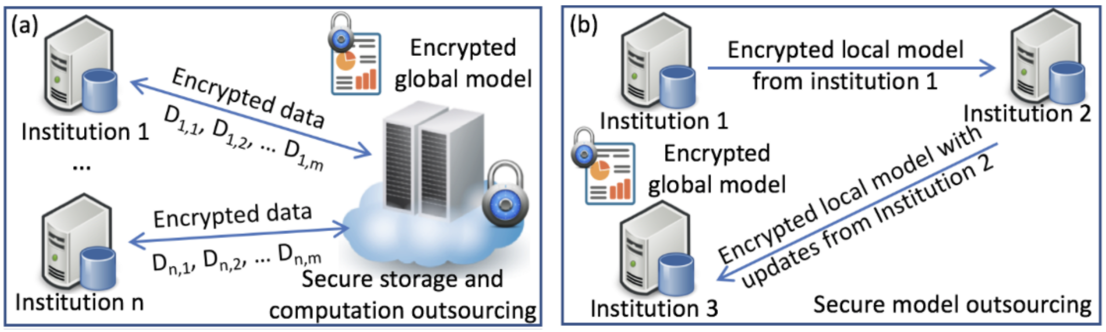

(a)
Institution 1
...
Institution n
Encrypted data
$D_{1,1}, D_{1,2}, \dots D_{1,m}$
Encrypted data
$D_{n,1}, D_{n,2}, \dots D_{n,m}$
Encrypted global model
Secure storage and computation outsourcing
(b)
Institution 1
Encrypted local model from institution 1
Institution 2
Encrypted global model
Encrypted local model with updates from institution 2
Institution 3
Secure model outsourcing

**Fig. 1.** Two secure models: (a) secure storage and computation outsourcing, (b) secure model outsourcing

vulnerabilities in sharing even the summary statistics; for example, the difference in mean age between a cohort of $n$ patients and another cohort of $n-1$ overlapped patients can reveal the actual age of a single patient.

Many medical decision-making systems rely on the logistic regression model [28, 9, 29]. However, to use them appropriately, we need to provide a sufficient sample, which requires a sample size calculation. Peduzzi et al. [25] suggested a simple guideline for a minimum number of cases to include in the study: let $p$ be the smallest of the proportions of negative or positive cases in the population and $k$ the number of covariates (the number of independent variables), then the minimum number of cases to include is $N = 10 \cdot k/p$. For example, one has three covariates to be included in the model and the proportion of positive cases in the population is 0.2 (20%). The minimum number of cases required is $10 \cdot 3/0.20 = 150$. For rare disease studies with many variables, it is even harder to collect enough samples from a single institution to meet this goal. We need to circumvent the privacy barriers to feed the model with more samples from different sources. As shown in Fig. 1, homomorphic encryption (HE) techniques can support typical secure computations (e.g. secure outsourcing and secure multiparty computation) and mitigate the privacy risks by allowing all computation to be done in the encrypted format.

Graepel et al. [14] shed light on machine learning with homomorphically encrypted data. The article discussed scenarios that are appropriate and inappropriate to exercise machine learning with homomorphic encryption techniques. The authors provided two examples: linear means classifier and linear discriminative analysis, which can be achieved by using low-degree polynomials in homomorphic encryption. However, these simple parametric models do not handle complex datasets well and they do not represent the mainstream machine learning technologies used in biomedical research [10, 21]. Additional work was carried out by Bos et al. [3] to demonstrate the feasibility of making a prediction on encrypted medical data in Microsoft's Azure cloud. However, instead of learning from the data, this model only makes predictions using learned logistic regression models in a privacy-preserving manner. Similarly, a more recent work called CryptoNets applied trained neural networks to encrypted data only for the evaluation purpose [13]. Related works are summarized in Table 1.

In the current literature, most similar to our work are Aono et al. [1] and Mohassel et al. [22], but they are also very different from ours in assumptions and methods. Aono et al. [1] introduced an approximation to convert the likelihood function into a low-degree polynomial and used an additive homomorphic encryption to aggregate some intermediary statistics. However, their scenario relies on the client to decrypt these intermediary statistics so that it can minimize the parameters locally. This is not a completely secure outsourcing scenario as ours, which works on encrypted data to obtain encrypted parameters without any client involvement. Mohassel et al. [22] developed secure two-party computation protocols

**Table 1.** Research works in secure analysis

| Reference | Problem | Techniques |
|---|---|---|
| Graepel et al. [14] | Linear means classifier/ Discriminative analysis | HE |
| Bos et al. [3] | Prediction using learned logistic regression model | HE |
| Gilad-Bachrach et al. [13] | Prediction using learned neural networks | HE |
| Aono et al. [1] | Logistic regression | Additive HE |
| Mohassel et al. [22] | Logistic regression | MPC |
| This work | Logistic regression | HE |

to conduct the stochastic gradient descent for solving logistic regression and neural network problems.

This method takes a completely different approach (garbled circuit and secret sharing vs homomorphic encryption) and the assumptions are widely different from ours (secure multiparty computation vs secure outsourcing). There are several prominent challenges related to scalability and efficiency. Traditional methods cannot handle many iterations of multiplications, which leads to a deep circuit and exponential growth in computational cost and storage size of the ciphertext. On the other hand, it is a nontrivial task to approximate certain critical functions in machine learning models using only low-degree polynomials. Naive approximation may lead to big errors and makes the solutions intractable. Our framework proposes novel methods to handle these challenges and makes it possible to learn a logistic regression model on encrypted data based completely on homomorphic encryption.

## 2 Backgrounds

In this section, we introduce the setting of our problem, followed by the homomorphic encryption scheme.

### 2.1 Logistic Regression

Logistic regression is a widely used learning model in biomedicine [10]. Data for supervised learning consist of pairs $(\boldsymbol{x}_i, y_i)$ of a vector of co-variates $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{id}) \in \mathbb{R}^d$ and a class label $y_i$ for $i = 1, \cdots, n$. We assume that $y_i \in \{\pm 1\}$ for binary classification. The model looks like:

$$\Pr[y_i | \boldsymbol{x}_i] = \sigma(y_i \cdot (1, \boldsymbol{x}_i)^T \boldsymbol{\beta})$$

for the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$ where $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \cdots, \beta_d)$ are the model parameters to be estimated. Training methods of logistic regression aim to find the optimal parameters $\boldsymbol{\beta}$, which minimizes the cost (negative log-likelihood)

$$\frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \cdot (1, \boldsymbol{x}_i)^T \boldsymbol{\beta})).$$

### 2.2 Homomorphic Encryption for Approximate Arithmetic

Homomorphic encryption is an encryption technique that allows computations on ciphertexts and generates encrypted results that match those of plaintext computation. This technology has great potentials in real-world applications since it allows us to securely outsource expensive computation on public (untrusted) server. The security of practical HE schemes [5, 11, 2] is based on the hardness of ring learning with errors (RLWE) problem. The inherent limitation of these constructions is that they only support the arithmetic operations over modular spaces so the required size of parameter for real number operations (i.e., no modular reduction over plaintext space) is too large to be practically implemented. On the other hand, real-world applications such as statistical testing, neural networks, and machine learning do not require absolute precision and they are all to a certain degree approximate.

We adopted a special cryptosystem developed by Cheon et al. [6], which supports an *approximate arithmetic* of encrypted messages. Different from existing methods, this cryptosystem trades precision for efficiency so that the size of parameters does not grow too large (thus computationally feasible). The cryptosystem supports key generation, encryption, decryption, addition, and multiplication operations. It also supports message packing and rotation, which are important to parallelize similar tasks.

The main idea is to consider an encryption noise (on the ciphertext for security) as a part of computation error occurring during approximate arithmetic. In other words, given a secret key $sk$, an encryption $\mathbf{c}$ of a plaintext $m$ satisfies the equation $\langle \mathbf{c}, sk\rangle = m + e \pmod q$ for some small error $e$.

We suppose that the underlying HE scheme is based on the $\mathsf{RLWE}$ assumption over the cyclotomic ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ for $N$ being a power of two. Let us denote by $[\cdot]_q$ the reduction modulo $q$ into the interval $(-q/2, q/2] \cap \mathbb{Z}$ of the integer. We write $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ for the residue ring of $\mathcal{R}$ modulo an integer $q$. The following is a description of the HE scheme [6].

- $\mathsf{ParamsGen}(\lambda)$: Given the security parameter $\lambda$, choose a power of two integer $N$, a modulus $Q = q^2$, and a discrete Gaussian distribution $\chi$. The $\mathsf{RLWE}$ problem of parameter $(N, Q, \chi)$ should achieve at least $\lambda$ bits of security level for the semantic security of cryptosystem. Output $params \leftarrow (N, q, \chi)$.
- $\mathsf{KeyGen}(params)$. Generate a polynomial $s \in \mathcal{R}$ by sampling its coefficient vector randomly from a sparse distribution on $\{0, \pm 1\}^N$ and set the secret key as $sk \leftarrow (1, s)$. Sample a polynomial $a$ uniformly at random from $\mathcal{R}_q$ and an error polynomial $e$ from $\chi$. Set the public key as $pk \leftarrow (b, a) \in \mathcal{R}_q \times \mathcal{R}_q$ where $b \leftarrow -as + e \pmod q$. Let $s' \leftarrow s^2$. Sample a polynomial $a'$ uniformly at random from $\mathcal{R}_Q$ and $e'$ from $\chi$. Set the evaluation key as $evk \leftarrow (b', a') \in \mathcal{R}_Q \times \mathcal{R}_Q$ where $b' \leftarrow -a's + e' + qs' \pmod Q$.
- $\mathsf{Enc}_{pk}(m)$. Sample a small polynomial $v$ (with $0, \pm 1$ coefficients) and two error polynomials $e_0, e_1$ from $\chi$. Output the ciphertext $v \cdot pk + (m + e_0, e_1) \in \mathcal{R}_q \times \mathcal{R}_q$.
- $\mathsf{Dec}_{sk}(\mathbf{c})$. For $\mathbf{c} = (c_0, c_1)$, output $m \leftarrow c_0 + c_1 \cdot s \pmod{q'}$.
- $\mathsf{Add}(\mathbf{c}, \mathbf{c}')$. Add two ciphertext and output $\mathbf{c}_{\mathsf{add}} \leftarrow \mathbf{c} + \mathbf{c}' \pmod{q'}$.
- $\mathsf{Mult}(\mathbf{c}, \mathbf{c}')$. For two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c}' = (c_0', c_1')$, let $(d_0, d_1, d_2) = (c_0c_0', c_0c_1' + c_1c_0', c_1c_1') \pmod{q'}$. Output $(d_0, d_1) + \left\lfloor \frac{1}{q} d_2 \cdot evk \right\rceil \pmod{q'}$.

The native plaintext space is simply represented as the set of polynomials in $\mathcal{R}$ with coefficient less than $q$, but it can be understood as a vector of $N/2$-dimensional complex vector using an encoding/decoding method described in [6] (each factor is called a *plaintext slot*). The SIMD technique enables to parallelize both space and computation time as in BGV scheme [12]. Addition and multiplication in $\mathcal{R}$ correspond the component-wise addition and multiplications on the plaintext slots. In addition, we could get a shift of the plaintext slots, that is, if $\mathbf{c}$ encrypts a plaintext vector $(m_1, m_2, \ldots, m_\ell)$, we could obtain the encryption of the shifted vector $(m_{k+1}, m_{k+2}, \ldots, m_\ell, m_1, \ldots, m_k)$. Let us denote the operation by $\mathsf{Rot}(\mathbf{c}; k)$.

A unique property of this cryptosystem is the following rescaling procedure, which plays an important role in controlling the magnitude of messages, and therefore, achieving the efficiency of approximate computation. The rescaling procedure coverts an encryption $\mathbf{c}$ of a message $m$ with a ciphertext modulus $q$ into an encryption $\mathbf{c}'$ of $r^{-1} \cdot m$ with the same secret key but a smaller modulus $q' = r^{-1} \cdot q$, in which $r$ is a scaling factor. We denote the output ciphertext by $\mathsf{RS}(\mathbf{c}; r)$. It enables us to round the message and reduce the size of significand by removing some inaccurate least significant bits as in the floating-point arithmetic. Informally, we will say that the input ciphertext modulus is reduced by $\log r$ bits after this procedure where the binary logarithm will be simply denoted by $\log(\cdot)$.

## 3 Secure Logistic Regression based on Homomorphic Encryption

Unlike linear regression, logistic regression does not have a closed form solution in most cases. As a result, we need to use nonlinear optimization methods to find the maximum likelihood estimators of the regression parameters. The Newton-Raphson [34] and the gradient descent [4] are the most commonly used methods for training. Because the Newton-Raphson method involves matrix inversion and most HE schemes do not naturally support division or matrix inversion, it is difficult to evaluate the method with HE schemes. On the other hand, gradient descent does not require the division operation, and therefore, it is a better candidate for homomorphically encrypted logistic regression. Thus, we choose the gradient descent algorithm as the training method for logistic regression.

Let $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \{\pm 1\}$ be the supervised learning samples for $i = 1, \cdots, n$. If we write $\boldsymbol{z}_i = y_i \cdot (1, \boldsymbol{x}_i) \in \mathbb{R}^{d+1}$, then the cost function for logistic regression is defined by $\frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-\boldsymbol{z}_i^T \boldsymbol{\beta}))$. Its gradient with respect to $\boldsymbol{\beta}$ is computed by $-\frac{1}{n}\sum_{i=1}^{n} \sigma(-\boldsymbol{z}_i^T \boldsymbol{\beta}) \cdot \boldsymbol{z}_i$. To find a local minimum point, the gradient descent method updates the regression parameters using the following formula until $\boldsymbol{\beta}$ converges:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \frac{\alpha}{n} \sum_{i=1}^{n} \sigma(-\boldsymbol{z}_i^T \boldsymbol{\beta}) \cdot \boldsymbol{z}_i,$$

where $\alpha$ is the learning rate.

### 3.1 Least Squares Approximation of the Sigmoid Function

Although the gradient descent method seems better suited than other training methods for homomorphic evaluation, some technical problems remain for implementation. In the preceding update formula, the sigmoid function is the biggest obstacle for evaluation, since the existing HE schemes only allow evaluation of polynomial functions. Hence the Taylor polynomials $T_d(x) = \sum_{k=0}^{d} \frac{f^{(k)}(0)}{k!} x^k$ have been commonly used for approximation of the sigmoid function ([3, 22]):

$$\sigma(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{48}x^3 + \frac{1}{480}x^5 - \frac{17}{80640}x^7 + \frac{31}{1451520}x^9 + O(x^{11}).$$

However, we observed the input values $\boldsymbol{z}_i^T \boldsymbol{\beta}$ of the sigmoid function during iterations on real-world datasets and concluded that the Taylor polynomial $T_9(x)$ of degree 9 is still not enough to obtain the desired accuracy (see Fig. 2). The size of error grows rapidly as $|x|$ increases. For instance, we have $T_9(4) \approx 4.44$, $T_9(6) \approx 31.23$, and $T_9(8) \approx 138.12$. In addition, we have to use a higher degree Taylor polynomial to guarantee the accuracy of regression, but it requires too many homomorphic multiplications to be practically implemented. In summary, the Taylor polynomial is not a good candidate for approximation because it is a *local approximation* near a certain point. Therefore, we adopted a *global approximation* method that minimizes the *mean squared error* (MSE). For an integrable function $f$, its mean square over an interval $I$ is defined by $\frac{1}{|I|}\int_I f(x)^2 dx$, where $|I|$ denotes the length of $I$. The least squares method aims to find a polynomial $g(x)$ of degree $d$ which minimizes the MSE $\frac{1}{|I|}\int_I (g(x) - f(x))^2 dx$. The least squares approximation has a closed formula that can be efficiently calculated using linear algebra.

**Fig. 2.** Graphs of sigmoid function and Taylor polynomials

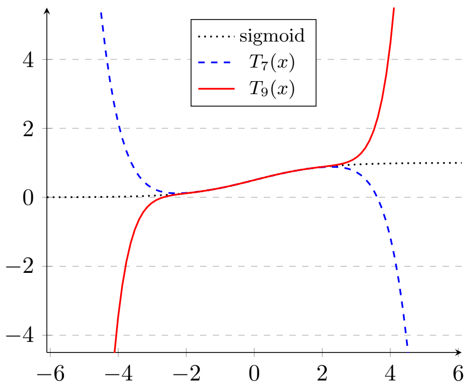
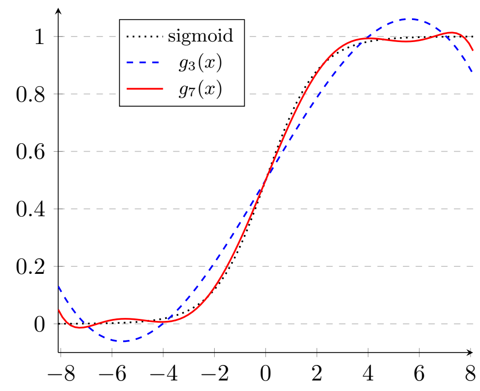


**Fig. 3.** Graphs of sigmoid function and least squares approximations



In our implementation, we used the degree 3 and 7 least squares approximations of the sigmoid function over the interval $[-8, 8]$, which contains all of the input values $(-\boldsymbol{z}_i^T \boldsymbol{\beta})$ during iterations. The least square polynomials are computed as:

$$g_3(x) = a_0 + a_1(x/8) + a_3(x/8)^3,$$
$$g_7(x) = b_0 + b_1(x/8) + b_3(x/8)^3 + b_5(x/8)^5 + b_7(x/8)^7,$$

where the coefficients vectors are $(a_0, a_1, a_3) \approx (0.5, 1.20096, -0.81562)$ and $(b_0, b_1, b_3, b_5, b_7) \approx (0.5, 1.73496, -4.19407, 5.43402, -2.50739)$. The degree 3 least squares approximation requires a smaller depth for evaluation while the degree 7 polynomial has a better precision (see Fig. 3).

## 3.2 Homomorphic Evaluation of Gradient Descent Algorithm

We will describe how to encode data and explain how to analyze logistic regression on encrypted data. To speed up the computation, we will use the packing mechanism to batch $n$ slots and perform $n$ evaluations in parallel, where $n$ is the number of training data samples.

We start with a useful aggregation operation across plaintext slots from the literature [15, 7, 8]. Specifically, given a ciphertext representing a plaintext vector $(m_1, \cdots, m_\ell)$, we introduce an algorithm (denoted by $\mathsf{AllSum}$) that generates a ciphertext representing a value of $\sum_{i=1}^{\ell} m_i$ in each plaintext slot. Assume that $\ell$ is chosen as a power-of-two integer. The cyclic rotation by one unit produces a ciphertext encrypting the plaintext vector $(m_2, \cdots, m_\ell, m_1)$. Then an encryption of the vector $(m_1 + m_2, m_2 + m_3, \cdots, m_\ell + m_1)$ is obtained by adding the original ciphertext. We repeatedly apply this method $(\log \ell - 1)$ times with a rotation by a power of two, which generates the desired ciphertext: that is, every plaintext slot contains the same value of $\sum_{i=1}^{\ell} m_i$. The $\mathsf{AllSum}$ algorithm is explicitly described in Algorithm 1.

**Algorithm 1** $\mathsf{AllSum}(\mathsf{ct})$

```
Input: Ciphertext ct encrypting plaintext vector $(m_1, \cdots, m_\ell)$
Output: Ciphertext ct encrypting $\sum_{i=1}^{\ell} m_i$ in each plaintext slot
1: for $i = 0, 1, \ldots, \log \ell - 1$ do
2:     Compute $\mathsf{ct} \leftarrow \mathsf{Add}(\mathsf{ct}, \mathsf{Rot}(\mathsf{ct}; 2^i))$
3: end for
4: return ct
```

Let us assume that we are given $n$ training data samples $\boldsymbol{z}_i$ with $(d+1)$ features. As mentioned before, our goal is to securely evaluate the following arithmetic circuit:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \frac{\alpha}{n} \sum_{i=1}^{n} g(-\boldsymbol{z}_i^T \boldsymbol{\beta}) \cdot \boldsymbol{z}_i, \tag{1}$$

where $g(x)$ denotes the approximating polynomial of the sigmoid function chosen in the previous subsection. We set the initial $\boldsymbol{\beta}$ parameters as the zero vector for simplicity.

Because our cryptosystem only supports integer computation, all the elements are scaled by a factor of an integer $p$ and then converted into the nearest integers for quantization. The client first receives the ciphertexts encrypting the vector $p \cdot \boldsymbol{z}_i$ from $n$ users, and then compromises them to obtain $(d+1)$ ciphertexts $\mathsf{ct.z}_j$ for $j = 0, 1, \dots, d$, each of which encrypts the vector $p \cdot (z_{1j}, \cdots, z_{nj})$ of the $j$-th attributes using batching technique. If $n$ is not a power of two, the plaintext slots are zero padded so that the number of slots divides $N/2$. Finally, these resulting ciphertexts $(\mathsf{ct.z}_0, \cdots, \mathsf{ct.z}_d)$ are sent to the the server for the computation of gradient descent.

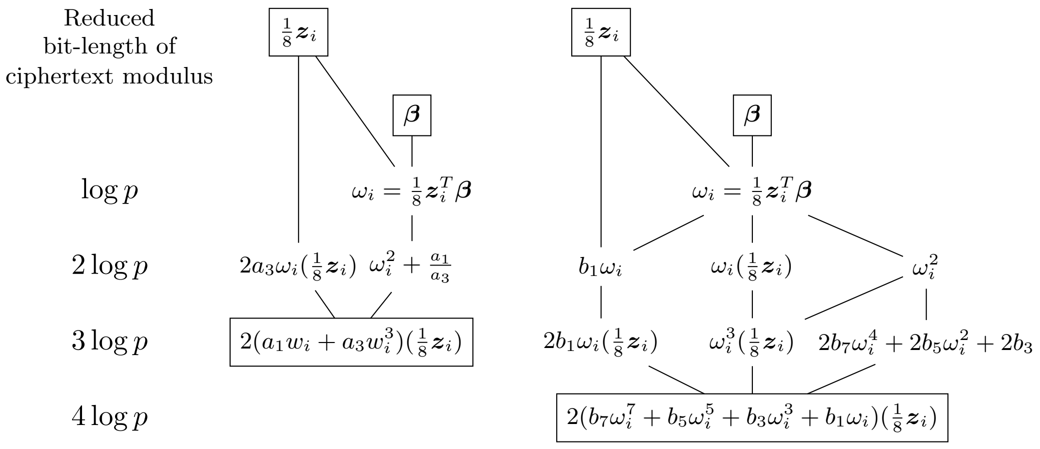
The public server generates the trivial ciphertexts $(\mathsf{ct.beta}_0, \cdots, \mathsf{ct.beta}_d)$ as zero polynomials in $\mathcal{R}_q$. At each iteration, it performs a homomorphic multiplication of ciphertexts $\mathsf{ct.beta}_j$ and $\mathsf{ct.z}_j$, and outputs a ciphertext encrypting the plaintext vector $p^2(z_{1j}\beta_j, \dots, z_{nj}\beta_j)$ for $j = 0, 1, \dots, d$. Then it aggregates the ciphertexts and performs the rescaling operation with a scaling factor of $p$ to manipulate the size of plaintext, returning a ciphertext $\mathsf{ct.ip}$ that represents a plaintext vector approximating to $p(\boldsymbol{z}_1^T\boldsymbol{\beta}, \dots, \boldsymbol{z}_n^T\boldsymbol{\beta})$.

For the evaluation of the least squares polynomial $g(x)$ at $(-\boldsymbol{z}_i^T\boldsymbol{\beta})$, we adapt the polynomial evaluation algorithm, denoted by $\mathsf{PolyEval}(\cdot)$, suggested in [6]. Each coefficient of the polynomial should be scaled by a factor of $p$ to be transformed into an integral polynomial. The output ciphertext $\mathsf{ct.g}$ contains $p \cdot g(-\boldsymbol{z}_i^T\boldsymbol{\beta})$ in the $i$-th slot. Finally, the server performs a homomorphic multiplication of the ciphertexts $\mathsf{ct.g}$ and $\mathsf{ct.z}_j$, $\mathsf{AllSum}$ procedure, and rescaling by a factor of $\lfloor \frac{n}{\alpha} \rceil$. These procedures generate the ciphertexts $\mathsf{ct.grad}_0, \dots, \mathsf{ct.grad}_d$ corresponding to the entries of the gradient vector weighted by the learning rate and the sample size. Then it only needs to perform an addition with the model parameters $\boldsymbol{\beta}$ and the gradient vector over encryption, which yields a new ciphertext $\mathsf{ct.beta}_j$ that encrypts an approximation of the $j$-th scaled value of the gradient update in Equation 1. Our secure logistic regression algorithm is described in Algorithm 2.

**Algorithm 2** Procedure of secure logistic regression algorithm

```
Input: Ciphertexts $\{\mathsf{ct.z}_j\}_{0 \le j \le d}$, a polynomial $g(x)$, and number of iterations $\mathsf{IterNum}$
 1: for $j = 0, 1, \dots, d$ do
 2:     $\mathsf{ct.beta}_j \leftarrow \mathbf{0}$
 3: end for
 4: for $i = 1, 2, \dots, \mathsf{IterNum}$ do
 5:     $\mathsf{ct.ip} \leftarrow \mathsf{RS}(\sum_{j=0}^{d} \mathsf{Mult}(\mathsf{ct.beta}_j, \mathsf{ct.z}_j); p)$
 6:     $\mathsf{ct.g} \leftarrow \mathsf{PolyEval}(-\mathsf{ct.ip}, \lfloor p \cdot g(x) \rceil)$
 7:     for $j = 0, 1, \dots, d$ do
 8:         $\mathsf{ct.grad}_j \leftarrow \mathsf{RS}(\mathsf{Mult}(\mathsf{ct.g}, \mathsf{ct.z}_j); p)$
 9:         $\mathsf{ct.grad}_j \leftarrow \mathsf{RS}(\mathsf{AllSum}(\mathsf{ct.grad}_j); \lfloor \frac{n}{\alpha} \rceil)$
10:         $\mathsf{ct.beta}_j \leftarrow \mathsf{Add}(\mathsf{ct.beta}_j, \mathsf{ct.grad}_j)$
11:     end for
12: end for
13: return $(\mathsf{ct.beta}_0, \dots, \mathsf{ct.beta}_d)$.
```

**Fig. 4.** Evaluation procedure of least squares approximations $(2g(\boldsymbol{z}_i^T\boldsymbol{\beta})-1)\cdot(\frac{1}{8}\boldsymbol{z}_i)$ when $g(x)=g_3(x)$ (left) or $g(x)=g_7(x)$ (right)



Reduced bit-length of ciphertext modulus
$\log p$
$2\log p$
$3\log p$
$4\log p$
$\frac{1}{8}\boldsymbol{z}_i$
$\boldsymbol{\beta}$
$\omega_i=\frac{1}{8}\boldsymbol{z}_i^T\boldsymbol{\beta}$
$2a_3\omega_i(\frac{1}{8}\boldsymbol{z}_i)$
$\omega_i^2+\frac{a_1}{a_3}$
$2(a_1w_i+a_3w_i^3)(\frac{1}{8}\boldsymbol{z}_i)$
$b_1\omega_i$
$\omega_i(\frac{1}{8}\boldsymbol{z}_i)$
$\omega_i^2$
$2b_1\omega_i(\frac{1}{8}\boldsymbol{z}_i)$
$\omega_i^3(\frac{1}{8}\boldsymbol{z}_i)$
$2b_7\omega_i^4+2b_5\omega_i^2+2b_3$
$2(b_7\omega_i^7+b_5\omega_i^5+b_3\omega_i^3+b_1\omega_i)(\frac{1}{8}\boldsymbol{z}_i)$

Our solution can compute the gradient descent algorithm securely; however, its direct implementation is not efficient and requires a total ciphertext modulus of $\log p\cdot(\lceil\log\deg g\rceil+3)+\lceil\log(\frac{n}{\alpha})\rceil$ bits at each iteration. We further optimized this algorithm by manipulating the arithmetic circuit for the update term $\frac{\alpha}{n}\sum_{i=1}^n g(-\boldsymbol{z}_i^T\boldsymbol{\beta})\cdot\boldsymbol{z}_i$ and could reduce the ciphertext modulus. We express the evaluation circuit as follows:

$$\boldsymbol{\beta}\leftarrow\boldsymbol{\beta}+\frac{4\alpha}{n}\sum_{i=1}^{n}\frac{\boldsymbol{z}_i}{8}-\frac{4\alpha}{n}\sum_{i=1}^{n}\big(2g(\boldsymbol{z}_i^T\boldsymbol{\beta})-1\big)\cdot\frac{\boldsymbol{z}_i}{8}. \tag{2}$$

If the client generates encryptions of $p(\frac{1}{8}\boldsymbol{z}_i)$ instead of $p\boldsymbol{z}_i$, the required bit length of ciphertext modulus per iteration can be reduced. On the other hand, the server uses a pre-computation step to reduce the complexity of update equation: it performs $\mathsf{AllSum}$ procedure and applies the rescaling operation with the scale factor of $\lfloor\frac{n}{4\alpha}\rceil$ on $\mathsf{ct.z}_j$ for all $j=0,1,\ldots,d$. As a result, we obtain a ciphertext $\mathsf{ct.sum}_j$ that encrypts an approximate value of $\frac{4\alpha p}{n}\sum_{i=1}^n\frac{z_{ij}}{8}$ in each plaintext slot. These ciphertexts will be stored during evaluation and used for update of the $j$-th component of weight vector $\boldsymbol{\beta}$. In particular, the ciphertexts $\mathsf{ct.beta}_0,\cdots,\mathsf{ct.beta}_d$ corresponding to the entries $\boldsymbol{\beta}$ becomes $\mathsf{ct.sum}_0,\cdots,\mathsf{ct.sum}_d$ at the first iteration.

Fig. 4 shows how to evaluate the arithmetic circuit $(2g(\boldsymbol{z}_i^T\boldsymbol{\beta})-1)\cdot(\frac{1}{8}\boldsymbol{z}_i)$ when $g(x)=g_3(x)$ or $g(x)=g_7(x)$. We take encryptions of $p\boldsymbol{\beta}$ and $\frac{p}{8}\boldsymbol{z}_i$ as inputs of the algorithm to minimize the number of required multiplications and depth. Consequently, the proposed method reduces the ciphertext modulus by $3\log p+\lceil\log(\frac{n}{4\alpha})\rceil$ bits (or $4\log p+\lceil\log(\frac{n}{4\alpha})\rceil$ bits) when $g(x)=g_3(x)$ (or $g(x)=g_7(x)$, respectively).

## 4 Implementations

In this section, we explain how to set the parameters and present our implementation results using the proposed techniques.

### 4.1 How to Set Parameters

It follows from Section 3.2 that a lower-bound on the bit length of a fresh ciphertext modulus, denoted by $\log q$, is as follows:

$$\begin{cases}\lceil\log(\frac{n}{4\alpha})\rceil+(\mathsf{IterNum}-1)(\lceil\log(\frac{n}{4\alpha})\rceil+3\log p)+\log q_0 & \text{when } g(x)=g_3(x),\\ \lceil\log(\frac{n}{4\alpha})\rceil+(\mathsf{IterNum}-1)(\lceil\log(\frac{n}{4\alpha})\rceil+4\log p)+\log q_0 & \text{when } g(x)=g_7(x),\end{cases}$$

where IterNum is the number of iterations of the gradient descent algorithm and $q_0$ is the output ciphertext modulus. The final ciphertext represents the desired vector $\boldsymbol{\beta}$ but is scaled by a factor of $p$, which means that $\log q_0$ should be larger than $\log p$.

The security of the underlying HE scheme relies on the hardness of the RLWE assumption. We derive a lower-bound on the ring dimension as $N \geq \frac{\lambda+110}{7.2} \cdot \log Q$ to get $\lambda$-bit security level from the security analysis of [12]. In other words, we will take the smallest integer $N$ that is a power-of-two integer satisfying this inequality.

## 4.2 Implementation Details

*Experimentation Environment.* All the experiments were performed on an Intel Xeon running at 2.3 GHz processor with 16 cores and 64GB of RAM, which is an m4.4xlarge AWS EC2 instance. In our implementation, we used a variant of a fixed-point homomorphic encryption scheme of Cheon et al. [6] with C++ based Shoup's NTL library [27]. Our implementation is publicly available at github [16].

*Datasets.* We develop our approximation algorithm using the Myocardial Infarction dataset from Edinburgh [18]. The others were obtained from Low Birth Weight Study (lbw), Nhanes III (nhanes3), Prostate Cancer Study (pcs), and Umaru Impact Study datasets (uis) [19, 23, 24, 31]. All these datasets have a single binary outcome variable, which can be readily used to train binary classifiers like logistic regression. Table 2 illustrates the datasets with the number of observations (rows) and the number of features (columns), respectively. We utilized five-fold cross-validation (CV) that randomly partitions the original datasets into five folds with the approximately equal size; we used four subsets for learning (with the learning rate $\alpha \approx 1$) and one subset for testing the trained model.

**Table 2.** Description of datasets

| Dataset | Number of observations | Number of features |
|---|---|---|
| Edinburgh | 1253 | 10 |
| lbw | 189 | 10 |
| nhanes3 | 15649 | 16 |
| pcs | 379 | 10 |
| uis | 575 | 9 |

*Parameters for the HE Scheme.* In our implementation, each coefficient of the secret key is chosen at random from $\{0, \pm 1\}$ and we set the number of nonzero coefficients in the key at $h = 64$. We used the standard deviation $\sigma = 3.2$ for a discrete Gaussian distribution to sample random error polynomials. We assumed that all the inputs had $\log p = 28$ bits of precision and set the bit length of the output ciphertext modulus as $\log q_0 = \log p + 10$. As discussed before, when evaluating the gradient descent algorithm with $g(x) = g_7(x)$, a ciphertext modulus is reduced more than $g(x) = g_3(x)$ at each iteration. Thus, we set the number of iterations as IterNum $= 25$ (resp. IterNum $= 20$) when $g(x) = g_3(x)$ (resp. $g(x) = g_7(x)$) to take an initial ciphertext modulus of similar size. We could actually obtain the approximate bit length of fresh ciphertext modulus ($\log q$) around 2204 to 2406. We took the ring dimension $N = 2^{17}$ to ensure 80 bits of security. Since all the computations were performed on encrypted data, the security against a semi-honest adversary follows from the semantic security of the underlying HE scheme. For this setting, the public key and a freshly encrypted ciphertext have two ring elements in $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^N + 1)$, so the size is

$2N \log q \approx 75$ MB. The key generation takes about 56∼58 seconds and the encryption takes about 1.1∼1.3 seconds.

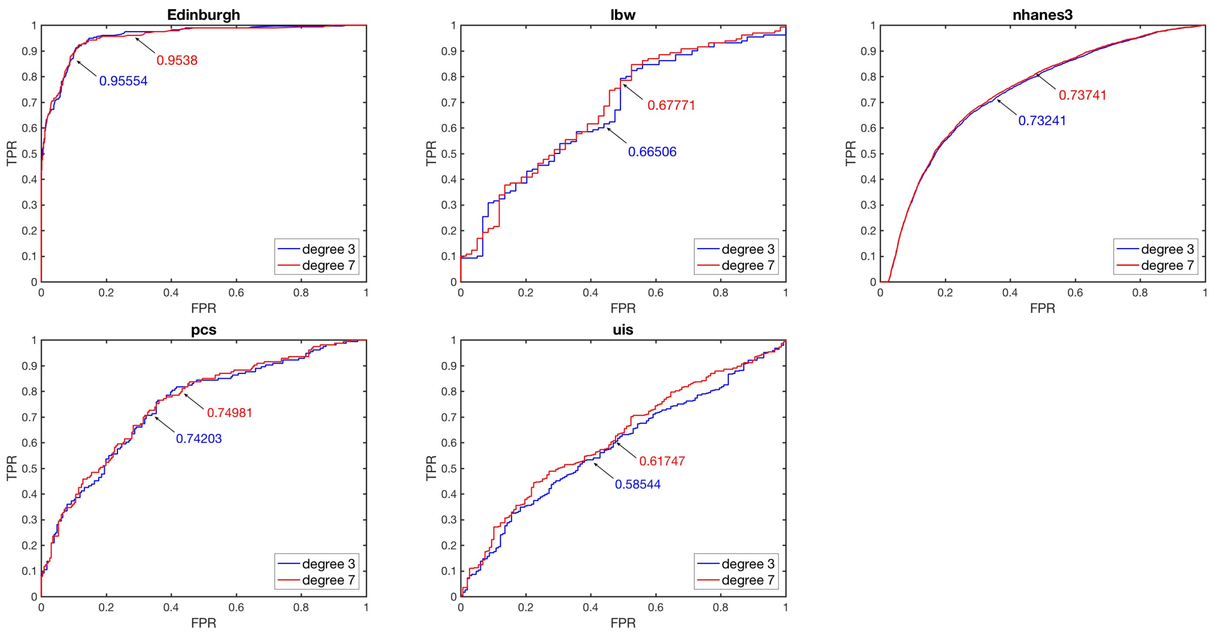
In Table 3, we evaluated our models performance based on average running time (encryption, evaluation, decryption) and storage (encrypted dataset size) in each fold.

**Table 3.** Experiment results of our HE-based logistic regression

| Dataset | Enc | deg($g$) | Eval | Dec | Storage |
|---|---|---|---|---|---|
| Edinburgh | 12s | 3 | 131min | 6.3s | 0.69GB |
| | | 7 | 116min | 6.0s | 0.71GB |
| lbw | 11s | 3 | 101min | 4.9s | 0.67GB |
| | | 7 | 100min | 4.5s | 0.70GB |
| nhanes3 | 21s | 3 | 265min | 12s | 1.15GB |
| | | 7 | 240min | 13s | 1.17GB |
| pcs | 11s | 3 | 119min | 4.4s | 0.68GB |
| | | 7 | 100min | 4.5s | 0.70GB |
| uis | 10s | 3 | 109min | 5.1s | 0.61GB |
| | | 7 | 94min | 4.3s | 0.63GB |

We used a popular metric, *Area Under the ROC Curve* (AUC), to measure the model's classification performance where the true positive rate (TPR) is plotted against the false positive rate (FPR) at various thresholds. Fig. 5 plots the average AUC values from five-fold CV for datasets. The program was implemented by MATLAB 2017a.

**Fig. 5.** Average AUC of encrypted logistic regression



We can converge to the optimum within a small number of iterations (20∼25), which makes it very promising to train a homomorphically encrypted logistic regression model and mitigate the privacy concerns.

**Table 4.** Comparison of encrypted/unencrypted logistic regression

| Dataset | IterNum | deg($g$) | HE-based LR | | Unencrypted LR | | MSE | NMSE |
|---|---|---|---|---|---|---|---|---|
| | | | Accuracy | AUC | Accuracy | AUC | | |
| Edinburgh | 25 | 3 | 86.03% | 0.956 | 88.43% | 0.956 | 0.0259 | 0.0261 |
| | 20 | 7 | 86.19% | 0.954 | 86.19% | 0.954 | 0.0007 | 0.0012 |
| lbw | 25 | 3 | 69.30% | 0.665 | 68.25% | 0.668 | 0.0083 | 0.0698 |
| | 20 | 7 | 69.29% | 0.678 | 69.29% | 0.678 | 0.0003 | 0.0049 |
| nhanes3 | 25 | 3 | 79.23% | 0.732 | 79.26% | 0.751 | 0.0033 | 0.0269 |
| | 20 | 7 | 79.23% | 0.737 | 79.23% | 0.737 | 0.0002 | 0.0034 |
| pcs | 25 | 3 | 68.85% | 0.742 | 68.86% | 0.750 | 0.0085 | 0.0449 |
| | 20 | 7 | 69.12% | 0.750 | 69.12% | 0.752 | 0.0002 | 0.0018 |
| uis | 25 | 3 | 74.43% | 0.585 | 74.43% | 0.587 | 0.0074 | 0.0829 |
| | 20 | 7 | 75.43% | 0.617 | 74.43% | 0.619 | 0.0004 | 0.0077 |

In Table 4, we compared the produced models using our encrypted approach and unencrypted logistic regression. In the unencrypted cases, we used the original sigmoid function on the same training dataset with the same iteration numbers as the encrypted cases. For discrimination, we calculated the accuracy (%), which is the percentage of the correct predictions on the testing dataset. For a more accurate comparison, we used the MSE that measures the average of the squares of the errors. We could also normalize it by dividing by the average of the squares of the (unencrypted) model parameters, called a *normalized mean-squared error* (NMSE).

## 5 Discussion

### 5.1 Principal Findings

Our implementation shows that the evaluation of the gradient descent algorithm with the degree 7 least squares polynomial yields better accuracy and AUC than degree 3. It is quite close to the unencrypted result of logistic regression using the original sigmoid function with the same number of iterations; for example, on the training model of Edinburgh dataset, we could obtain the model parameters $\boldsymbol{\beta}$ as follows: $\boldsymbol{\beta} = (-1.7086, 0.0768, 0.1119, 0.3209, 1.2033, 0.3684, 0.9756, 0.2020, 0.2259, -0.1641)$, which can reach 86.19% accuracy and 0.954 AUC on the testing dataset. When using the sigmoid function on the same training dataset, the model parameters $\boldsymbol{\beta}$ are $(-1.6308, 0.0776, 0.1097, 0.3155, 1.1809, 0.3651, 0.9599, 0.2083, 0.2298, -0.1490)$, which give the same accuracy and AUC. On the other hand, as shown in Table 4, the MSE and NMSE values of degree 7 are closer to zero, which inspires us that the polynomial approximation of that degree is fairly accurate for logistic regression.

One of the inherent properties of our underlying HE scheme is that the inserted errors for security may increase after some homomorphic operations. Hence, the size of error and the precision loss should be discussed carefully to guarantee the correctness of the resulting value. On the other hand, the gradient descent method has a property of negative feedback on computational error. Because we use the gradient at the current weight vector $\boldsymbol{\beta}$ to move it closer to the optimal point of minimized cost, the effect of noise disappears after some iterations. Therefore, there is no need to manage the precision of messages to confirm the correctness of resulting value because the noises are not amplified during evaluation. In our experimentation on the Edinburgh dataset, for instance, the difference between the model parameters obtained from encrypted/unencrypted evaluations was less than $2^{-11}$. This means that we can precisely compute at least most significant 11 bits after the radix point of the model parameters and this approximate vector is accurate enough to achieve a good performance in testing data samples.

### 5.2 Limitations

There are still a number of limitations in the application of our evaluation model to an arbitrary dataset. First of all, the use of HE yields the overheads in computation and storage. The size of dataset should be limited for practical evaluation, but this is not a big problem since there have been significant improvements in the existing HE schemes. The development of HE technology will achieve much better practical performance in our protocol.

Another issue arises from the polynomial approximation. We suggested the least squares method on a certain interval $[-8, 8]$, but the precision of result can increase by managing approximation error from wider range inputs. Finally, our model is based on fixed hyper parameters that should be decided before starting of the evaluation. It would be highly beneficial if we could detect convergence of the loss function in the training process and support early stop instead.

## 6 Conclusions

This paper presents the first effective methodology to evaluate the learning phase of logistic regression using the gradient descent method based on HE. We have demonstrated the capability of our model across the experiments with different biological datasets. In particular, our solution can be applied to a large-scale dataset, which shows the feasibility of our approach.

## Acknowledgements

The authors would like to thank Kristin Lauter for helpful discussions and suggestions. The authors would also like to thank Andrey Kim for extensive assistance with the code for the homomorphic encryption scheme. We also thank anonymous JMIR Medical Informatics reviewers for their careful reading of the paper and diligent comments.

The work of M. Kim, S. Wang, and X. Jiang was supported by the Patient-Centered Outcomes Research Institute (PCORI) under contract ME-1310-07058, the National Institute of Health (NIH) under award number R01HG008802, R01GM114612, R01GM118574, R01GM118609, R21LM012060, and U01EB023685. Y. Song was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) (No.2017R1A5A1015626).

## References

1. Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144. ACM, 2016.
2. J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding*, pages 45–64. Springer, 2013.
3. J. W. Bos, K. Lauter, and M. Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.
4. L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.
6. J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 409–437. Springer, 2017.
7. J. H. Cheon, M. Kim, and M. Kim. Search-and-compute on encrypted data. In *International Conference on Financial Cryptography and Data Security*, pages 142–159. Springer, 2015.
8. J. H. Cheon, M. Kim, and M. Kim. Optimized search-and-compute circuits and their application to query evaluation on encrypted data. *IEEE Transactions on Information Forensics and Security*, 11(1):188–199, 2016.
9. E. Dietz. Application of logistic regression and logistic discrimination in medical decision making. *Biometrical journal*, 29(6):747–751, 1987.
10. S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5):352–359, 2002.
11. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.
12. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, volume 7417, pages 850–867. Springer, 2012.
13. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.
14. T. Graepel, K. Lauter, and M. Naehrig. ML confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.
15. S. Halevi and V. Shoup. Algorithms in HElib. In *Advances in Cryptology–CRYPTO 2014*, pages 554–571. Springer, 2014.
16. HELR. Implementation of secure logistic regression based on homomorphic encryption. `https://github.com/K-miran/HELR`, 2017.
17. W. Jiang, P. Li, S. Wang, Y. Wu, M. Xue, L. Ohno-Machado, and X. Jiang. WebGLORE: a web service for Grid LOgistic regression. *Bioinformatics*, 29(24):3238–3240, 2013.
18. R. Kennedy, H. Fraser, L. McStay, and R. Harrison. Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: derivation and evaluation of logistic regression models. *European heart journal*, 17(8):1181–1191, 1996.
19. lbw. lbw: Low birth weight study data in logisticdx: Diagnostic tests for models with a binomial response. `https://rdrr.io/rforge/LogisticDx/man/lbw.html`, 2017.
20. Microsoft. Hosting and cloud study 2016. `http://download.microsoft.com/download/9/4/1/941D4C3D-5093-4468-BE11-FB18E4FBD199/Cloud%20and%20Hosting%20Trends%20-The%20Digital%20Revolution,%20Powered%20by%20Cloud.pdf`, 2016.
21. R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, page bbx044, 2017.
22. P. Mohassel and Y. Zhang. SecureML: A system for scalable privacy-preserving machine learning. IEEE Symposium on Security and Privacy, 2017.
23. nhanes3. Nhanes iii data in logisticdx: Diagnostic tests for models with a binomial response. `https://rdrr.io/rforge/LogisticDx/man/nhanes3.html`, 2017.
24. pcs. Prostate cancer study data in logisticdx: Diagnostic tests for models with a binomial response. `https://rdrr.io/rforge/LogisticDx/man/pcs.html`, 2017.
25. P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein. A simulation study of the number of events per variable in logistic regression analysis. *Journal of clinical epidemiology*, 49(12):1373–1379, 1996.
26. D. Rousseau. Biomedical research: Changing the common rule by david rousseau — ammon & rousseau translations. `https://www.ammon-rousseau.com/changing-the-rules-by-david-rousseau/`, 2017.
27. V. Shoup. Ntl: A library for doing number theory. `https://shoup.net/ntl/`, 2010.
28. E. R. Skinner, G. O. Barnett, D. E. Singer, A. G. Mulley, R. A. Lew, S. A. Stickler, M. M. Morgan, and G. E. Thibault. The use of logistic regression in diagnostic and prognostic prediction in a medical intensive care unit. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, volume 1, page 222. American Medical Informatics Association, 1980.
29. E. W. Steyerberg, M. J. Eijkemans, F. E. Harrell Jr, and J. D. F. Habbema. Prognostic modeling with logistic regression analysis: in search of a sensible strategy in small data sets. *Medical Decision Making*, 21(1):45–56, 2001.
30. J. J. Trinckes and Jr. *The Definitive Guide to Complying with the HIPAA/HITECH Privacy and Security Rules.* CRC Press, 3 Dec. 2012.

31. uis. Umaru impact study dat in logisticdx: Diagnostic tests for models with a binomial response. `https://rdrr.io/rforge/LogisticDx/man/uis.html`, 2017.
32. S. Wang, X. Jiang, Y. Wu, L. Cui, S. Cheng, and L. Ohno-Machado. Expectation Propagation LOgistic REgression (EXPLORER): distributed privacy-preserving online model learning. *Journal of biomedical informatics*, 46(3):480–496, 2013.
33. Y. Wu, X. Jiang, J. Kim, and L. Ohno-Machado. Grid Binary LOgistic REgression (GLORE): building shared models without sharing data. *Journal of the American Medical Informatics Association*, 19(5):758–764, 2012.
34. T. J. Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995.