3L1-OS-06a-3

Privacy-preservation for Stochastic Gradient Descent Application to Secure Logistic Regression

Shuang Wu[*1] Tadanori Teruya[*3] Junpei Kawamoto[*2] Jun Sakuma[*1] Hiroaki Kikuchi[*4]

[*1] Dept. of Computer Science, Graduate School of SIE, University of Tsukuba
[*2] Faculty of Engineering, Information and Systems, University of Tsukuba
[*3] Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology
[*4] School of Interdisciplinary Mathematical Sciences, Meiji University

Logistic regression is one of the most widely used statistical methods in the fields of medical, social sciences, marketing and etc. Enabling privacy preservation in logistic regression has attracted much attention recently. However, logistic function is not available in the secure settings (encryptions are not applicable) because of its non-linear property. In this work, we propose a preliminary approach to realize privacy-preserving logistic regression in cryptographic notion. We propose to approximate logistic function by polynomial fitting in order to make encryption applicable. And the experiment shows that our approach achieves good prediction accuracy compared with original logistic regression.

1. Introduction

The traditional paradigm in machine learning is to learn a decision model (such as a classifier) from a given data set. Many techniques in machine learning follow gradient descent method to discover this decision model, such as logistic regression, online kernel machines, and etc. These algorithms often assume free access to data, either at a centralized location or in federated forms. Nowadays, data are often distributed among multiple parties: financial data are often distributed among multiple banks and credit institutions; medical records are distributed among multiple hospitals and health care centers. Privacy and security concerns restrict these parties to directly share their private data with others. In order to construct efficient machine learning algorithms on the overall data while preserving privacy of each party's data, privacy-preserving machine learning becomes an emerging problem.

Logistic regression is a multivariate regression model used for measuring the relationship between a categorical dependent variable and several independent variables, by converting the dependent variable to probability scores. It is one of the most widely used statistical methods in social and medical science fields. For example, it might be used to predict whether a voter will vote Democratic or Republican, or whether a patient has disease. It is valuable to enable privacy-preservation in logistic regression. For privacy preservation in distributed machine learning, cryptographic protocols, which are specifically designed for the target algorithms, are often used. However, it is difficult to design cryptographic protocols for logistic model, because logistic function (sigmoid function), the crucial part in logistic model, is not readily available in the secure settings due to its non-linear property. There exists some works that incorporate privacy into logistic regression. Fienberg et al. [2] focus on " secure " logistic regression for horizontally partitioned databases; they used secure summation, which takes advantage of random perturbation, to fit with the exponential family formulation. Another work proposed by Chaudhuri et al. [1] considered a privacy-preserving method for logistic regression using the concept of $\epsilon-$differential privacy. All of these works avoid the problems of dealing with logistic function in the cryptographic model by the other exist privacy preserving data analysis, such as perturbation of the data and etc.

In our work, we propose cryptographically secure logistic regression following stochastic gradient descent on vertically partitioned databases. In order to alleviate the problem mentioned above, we make use of polynomial fitting to approximate the logistic function. Our approach achieves good prediction accuracy compared with original logistic regression.

We organize the paper as follows: Section 2 gives a brief introduction of logistic regression and homomorphic properties of cryptography. Section 3 presents the two party setup, polynomial fitting method and protocol for privacy-preserving logistic regression. In section 4, we illustrate the experiment and show the results.

Contact: Shuang Wu, Department of Computer Science, University of Tsukuba, 1-1-1 Tennodai Tsukuba Ibaraki Japan, 029-853-3826, anita_ ws[at] mdl.cs.tsukuba.ac.jp

2. Preliminaries

2.1 Gradient Descent Methods

Consider about a supervised learning setup. Each example is a pair of $(\mathbf{x}, y)$. Loss function $l(\hat{y}, y)$ that measures the cost of predicting $\hat{y}$ when the actual answer is y, and we consider the decision model f as $f(\mathbf{x})$ which is a differentiable function and parameterized by a weight vector $\mathbf{w}$. Let there be N examples in the data set, then the *empirical risk* of prediction function f is given by:

$$E_N(f) = \frac{1}{N}\sum_{i=1}^{N} l(f(\mathbf{x}_i), y_i)$$

Many techniques in machine learning follow the gradient descent methods[3] to learn the decision model. Batch Gradient Descent (BGD) method is often used to minimize this empirical risk, in

which each iteration updates **w** by the gradient of $E_N(f)$:

$$w_{t+1} = w_t - \eta \frac{1}{N} \nabla_w \sum_{i=1}^{N} l(f(\mathbf{x}_i), y_i)$$

In Stochastic Gradient Descent (SGD) [3], instead of computing the gradient of $E_N(f)$ exactly, the gradient is estimated on the basis of a single randomly picked example $(\mathbf{x}_t, y_t)$ at each iteration:

$$w_{t+1} = w_t - \eta_t \nabla_w l(f(\mathbf{x}_i), y_i)$$

SGD is a great simplification of BGD, and it is attractive in terms of computation time when the number of examples is huge. In our work, we use SGD to learn the classifier of logistic regression.

2.2 Logistic Regression

The logistic model is designed to describe a probability, which is always some number between 0 and 1. For example, in epidemiologic terms, such a probability gives the risk of an individual getting a disease. Therefore, it is set up to ensure that whatever estimate of risk we get, it will always be some number between 0 and 1. The logistic function (sigmoid function) is defined as:

$$f(\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T \mathbf{x})}$$

Let D be a set of N points:

$$D = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in R^m, y_i \in \{0, 1\}\}_{i=1}^{N}$$

where y_i, in the epidemiologic example again, is either 1 or 0. y_i is reffered to be the label, which indicates if a patient is diseased or non-diseased. The weight vector **w** is obtained by maximizing the loglikelihood:

$$\log l(\mathbf{w}) = \sum_{i=1}^{N} y_i \log f(\mathbf{x}_i) + (1 - y_i) \log(1 - f(\mathbf{x}_i))$$

The gradient has the form:

$$\nabla_{\mathbf{w}} = -\sum_{i=1}^{N} (y_i - f(\mathbf{w}, \mathbf{x}_i)) \mathbf{x}_i$$

As for stochastic gradient descent, the update process performs as follows:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \eta_t (y_i - f(\mathbf{w}, \mathbf{x}_i)) \mathbf{x}_i$$

The objective function of logistic regression is known to be convex. Thus, SGD procedure leads to the global optimal solution.

2.3 Pailliar Cryptosystem

The Paillier Cryptosystem [4] is a public key encryption scheme that can be used to conceal information, with its homomorphic properties.

- Homomorphic addition of plaintexts
  - $Enc_{pk}(m_1; r_1) Enc_{pk}(m_2; r_2) \bmod n^2$ $= Enc_{pk}(m_1 + m_2; r_1, r_2) \bmod n$
- Homomorphic multiplication of plaintexts
  - $Enc_{pk}(m_1; r)^{m_2} \bmod n^2 = Enc_{pk}(m_1 m_2; r) \bmod n$

In our work, we illustrate our proposal by using Pailliar cryptosystem to preserve privacy of data.

3. Privacy-preserving Logistic Regression

In this section, we propose a protocol to perform two party privacy preserving logistic regression by Pailliar cryptosystem. We also propose a sub-protocol to achieve secure polynomial computation.

3.1 Problem Statement

Classification includes two tasks: learning a classifier from data (training data) with class labels; predicting the class labels for unlabeled data by the classifier. For example, consider about a classification task of predicting diseases. The training data is made up of medical records with labels " diseased " or " non-diseased ". The classifier is used to predict whether the unlabeled medical records are diseased or non-diseased.

Realization of the diseases predicition work involves two parties: the health care center (Party A) who keeps the medicial records of individuals and the hospital (Party B) who keeps diagnosis information of patients.

Given a data set X which includes N vectors: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, each vector $\mathbf{x}_i$ represents medical records for an individual and target **y** as shown below:

$$X : \begin{array}{c} \mathbf{x}_1 = (x_{1,1} \quad x_{1,2} \quad \cdots \quad x_{1,m}) \\ \mathbf{x}_2 = (x_{2,1} \quad x_{2,2} \quad \cdots \quad x_{2,m}) \\ \vdots \qquad\quad \vdots \qquad\quad \ddots \qquad\quad \vdots \\ \mathbf{x}_N = (x_{N,1} \quad x_{N,2} \quad \cdots \quad x_{N,m}) \end{array}$$

$$\mathbf{y} : [y_1, y_2, \cdots, y_N]^T$$

We say that y_i is $\mathbf{x}_i^{'}s$ class label (or observed target value).

As the example above, here the $(1 \times m)$ vector $\mathbf{x}_i$ can be viewed as holding the information of each individual i's medical record, and y_i holds the corresponding diagnosis. So the health care center which will be represented by Party A holds X as its private information; the hospital, represented by Party B, holds vector **y** as the private information.

We are going to design a two party privacy-preserving protocol for logistic regression. As mentioned above, the logistic function is not readily available in the secure settings in which homomorphic property of encryption works only for linear models. So, the crucial part of enabling privacy-preserving logistic regression is finding appropriate methods to deal with logistic function. We propose to approximate logistic function by polynomial fitting.

3.2 Polynomial Fitting

The logistic model takes advantage of the logistic function (sigmoid function), which is not readily available in the secure settings. In order to keep the good property of logistic model, at meanwhile maintaining the privacy of data, we propose a method to approximate the logistic function in a way that the homomorphic property is available.

We aim to approximate the logistic function by polynomial fitting. For example, Figure 1 shows a 7th degree polynomial function $g(z) = \sum_{i=1}^{7} \alpha_i z^i$. From the plot, we can see that this polynomial function matches well with logistic function, however, there exists oscillation where $z \in [-\infty, -4] \cup [4, \infty]$. So, when using polynomial fitting to approximate logistic function, some assumptions should be made: when the output of the simulated polynomial function is not within the interval of $[0, 1]$, it will be forced

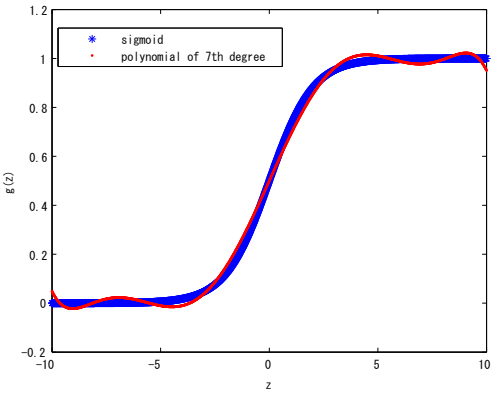

Figure 1: 7th polynomial fitting

to be within it. Thus, maintain the quality of logistic function (sigmoid function).

Using polynomial fitting for the logistic function, the whole process of logistic regression takes only sums and products, because of which we are able to maintain the validity of cryptography.

3.3 Protocol 1

We propose **Protocol 1** to perform two party secure computation of polynomial $g(z)$. It will be used as a sub-protocol for realizing privacy-preserving logistic regression.

$$g(z) = \sum_{i=1}^{k} \alpha_i z^i$$

In this protocol, the two party: Alice and Bob take integer $z - r$ and r respectively as their private input. During the secure computation, both parties know nothing but their own private information. And after secure computation, they share the value of polynomial $g(z)$.

As in Pailliar cryptosystem, all the messages to be encrypted must be integers, we expand all the coeffients $\alpha_i (1 \leqslant i \leqslant k)$ to the times of M (an integer that is large enough to make α_i to be integers).

For brevity, we eliminate random numbers form the encryption equation. We write $Enc_{pk}(m)$ instead of $Enc_{pk}(m; r)$ in the protocols.

3.4 Protocol 2

We propose **Protocol 2** to perform two-party secure logistic regression. In order to distinguish the two parties from **Protocol 1**, we here use A and B to represent the two parties. Party A takes N vectors from $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ as input, and Party B takes vector **y** as input. During the whole process, Party A knows nothing but its own information X,while Party B learns nothing but the weight vector and its own information. When the protocol terminates, Party B learns the final classifer **w**. Then, it can use the securely computed weight vector **w** to perform prediction.

In this protocol, step size η_t might not be integer along with the increasing of t. So, as in **Protocol 1**, we expand step size η_t to the times of $M^{'}$ (an integer that is large enough to make all the η_t to be integers).

Protocol 1 Secure Polynomial Computing

INPUT OF Alice: $z - r \in \mathbb{Z}_q \quad (q \ll n)$
INPUT OF Bob: $r \in \mathbb{Z}_q$
OUTPUT: shares $s_A + s_B = Mg(z) \in \mathbb{Z}_M$
Coefficient $M\alpha_i (1 \leqslant i \leqslant k)$ is publicly available
Setup: Alice does:
  generate a private and public key pair (sk, pk)
  send pk to Bob
1. Alice does:
  for i from 1 to k
    send $c_i = Enc_{pk}((z - r)^i)$ to Bob
2. Bob does:
  for i from 1 to k:
    compute $Enc_{pk}(z^i) = c_i \cdot \prod_{j=1}^{i} Enc_{pk}^{\theta_j r^j}(z^{i-j})$
  set $h \leftarrow \prod_{i=1}^{k} Enc_{pk}^{M\alpha_i}(z^i)$
  send $h^{'} = h \cdot Enc_{pk}(-s_B)$ to Alice
3. Alice computes $s_A = Dec_{sk}(h^{'}) = Mg(z) - s_B$
(θ_i is the opposite number of corresponding binomial coeffient, for example, 3rd degree of polinomial: $\theta_1 = 3$, $\theta_2 = -3$, $\theta_3 = 1$)

Security Analysis: this protocol securely updates **w** at each iteration, however some vulnerabilities remain. After two more iterations, Party B learns the difference of **w** at each update. If **x** is a low-dimensional binary vector, this may leak some information about **x**. We are trying to figure out effective methods to fix this problem. One way is updating **w** within the cypher forms, in other words, only the final **w** can be seen by Party B. This need us to detect some support protocols to handle division operation. Regarding the "difference attack on **w**", we give up describing more secure version due to the space limitation.

3.5 Computational Complexity

In this section, we derive the computational complexity of **Protocol 1** and **Protocol 2**.

We illustrate the computational complexity of **Protocol 1** as follows: In Step 1, Alice encrypts the messages for k times. If we define the computational complexity of encryption as $O(\tau(n))$ where n is the important parameter in Pailliar cryptosystem and $\tau(n)$ is an expression for computational complexity of n with respect to the encryption process. Hence, in this step, the computational complexity is $O(k\tau(n))$. In Step 2, Bob first computes the cypher text of z^i and set h, which make the computational complexity to be $O(\tau(n) + k^2 + k)$. In Step 3, computational complexity is $O(\tau(n))$, as Alice does decryption once. We summarize the computional complexity of this protocol as $O(g) = O((k + 2)\tau(n) + k^2 + k)$.

For **Protocol 2**, we determine the computational complexity of one iteration as follows: in Step 1, as Party A does encryptions for m times, the computational complexity is $O(m\tau(n))$. Step 2 takes $O(m + \tau(n))$ and Step 3 takes $O(\tau(n))$. In Step 4, two parties perform the secure polynomial computation. So the computational complexity is $O(g)$ as shown above. In Step 6, Party B generates random numbers and does the updation, which make the computational complexity to be $O(2m\tau(n)+2m)$. And Step 7 and 8 take $O(m\tau(n))$ and $O(m\tau(n) + m)$ respectively. The overall complexity for the entire process of **Protocol 2** is:

Protocol 2 Privacy-preserving Logistic Regression

INPUT OF A: Dataset $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{Z}_n^m$
INPUT OF B: Target vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)$, $y_i \in \{0, 1\}$
OUTPUT: final classifier $\mathbf{w} = (w_1, w_2, \ldots, w_m)$
(a) Setup: A generates a private and public key pair (sk, pk) and send pk to B; B generates a $(1 \times m)$ zero vector $\mathbf{w}$.
(b) A randomly shuffles or reorders the Dataset X, and for each element $\mathbf{x}_i (1 \leqslant i \leqslant N)$ of $\mathbf{X}$, the following is performed:

1. A does:
   for j from 1 to m:
   send $Enc_{pk}(x_{i,j})$ to B.
2. B does:
   generate random number: r
   send $c = Enc_{pk}(-r) \cdot \prod_{j=1}^{m} Enc_{pk}(x_{i,j})^{w_j}$ to A.
3. A does:
   compute $d = Dec_{sk}(c) = \mathbf{w} \cdot \mathbf{x}_i - r$
4. A and B conduct **Protocol 1**:
   A takes d and B takes r as input.
   A and B share $s_A + s_B = Mg(\mathbf{w} \cdot \mathbf{x}_i)$.
5. A does:
   for j from 1 to m:
   send $c_j^{'} = Enc_{pk}(s_A x_{ij})$ to B.
6. B does:
   for j from 1 to m:
   generate random number r_j
   compute $c_j^{''} = c_j^{'} Enc_{pk}(x_{ij})^{s_B}$
   send $c_j^{'''} =$
   $Enc_{pk}(w_j) \cdot Enc_{pk}(x_{ij})^{MM^{'}\eta y_i} \cdot c_j^{''-M^{'}\eta} \cdot Enc_{pk}(-r_j)$ to A.
7. A does:
   for j from 1 to m:
   send $d_j^{'} = Dec_{sk}(c_j^{'''})$ to B.
8. B does:
   compute $w_j^{'} = d_j^{'} + r_j$
   normalization $w_j = \dfrac{w_j^{'}}{MM'}$
   get $\mathbf{w} = (w_1, \ldots w_j, \ldots, w_m)$

$O(N \times ((4m + 2)\tau(n) + 5m + O(g)))$.

4. Experiment

In this experiment, we implement the original logistic regression and our approach. The prediction accuracy is adopted to verify that our approach maintain good performance while preserve the privacy.

4.1 Dataset

Dataset SPECT[*1] (23 attributes, 267 examples), which describes diagnosing of cardiac Single Proton Emission Computed Tomography images, is used in this experiment.

4.2 Settings

We implement the original logistic regression and our protocol. At present, our protocol is implemented without any communications for the purpose of testing the feasibility. We conduct 7th degree polynomial fitting to approximate logistic function as mentioned in Sec. 3.2.

∗1 http://archive.ics.uci.edu/ml/datasets/SPECT+Heart

Table 1: Average prediction accuracy

	Average Accuracy (%)
Original Algorithm	82.89
Our approach (without communication)	82.35

4.3 Result

Table 2. shows the average prediction accuracy of 100 times running of original logistic regression and our approach. Our approach achieves 82.35% prediction accuracy compared with the 82.89% of the original algorithm. This difference is aroused by the approximation of polynomial fitting. There exists some deviations between the polynomial function and the logistic function.

5. Conclusion

In this work, we have demonstrated a priliminary approach to achieve two party privacy-preserving logistic regression. We propose to approximate the logisitic function by polynomial fitting to handle the problem that logistic function is not readily available in the secure settings. Our approach achieves good prediction accuracy compared with the orginal logistic regression. Fixing the vulnerability of the protocol remains to be our future work.

Acknowledgement

This work was supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (A), Grant Number 12913388; and Grand-in Aid Reserch (B), Grant Number 22300028.

References

[1] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, pages 289–296. Curran Associates, Inc., 2008.

[2] Stephen E. Fienberg, William J. Fulp, Aleksandra B. Slavkovic, and Tracey A. Wrobel. "secure" log-linear and logistic regression analysis of distributed databases. In Josep Domingo-Ferrer and Luisa Franconi, editors, *Privacy in Statistical Databases*, volume 4302 of *Lecture Notes in Computer Science*, pages 277–290. Springer, 2006.

[3] Bottou L. Large-scale machine learning with stochastic gradient descent. *In Lechevallier, Y. and Saporta, G., editors, Compstat 2010,*, 2010.

[4] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.