A Privacy Preserving Deep Linear Regression Scheme Based on Homomorphic Encryption

Danping Dong[1, *], Yue Wu[1], Lizhi Xiong[1] and Zhihua Xia[1]

Abstract: This paper proposes a strategy for machine learning in the ciphertext domain. The data to be trained in the linear regression equation is encrypted by SHE homomorphic encryption, and then trained in the ciphertext domain. At the same time, it is guaranteed that the error of the training results between the ciphertext domain and the plaintext domain is in a controllable range. After the training, the ciphertext can be decrypted and restored to the original plaintext training data.

Keywords: Linear regression, somewhat homomorphic encryption, machine learning.

1 Introduction

As one of the artificial intelligence technologies, machine learning has been rapidly developed in recent years and widely used in medicine [Bonawitz, Ivanov, Kreuter et al. (2017); Zhang, Yang and Chen (2016); Phong, Aono, Hayashi et al. (2017)], Internet of things [Shokri and Shmatikov (2015)] and cyberspace security [Liu, Jiang, Chen et al. (2017); Li, Li, Huang et al. (2017)]. Among them, the linear regression technique is particularly suitable for regression problems because it can output a continuous value. Therefore, it is widely used in practical scenarios, such as predicting continuous values of house price, temperature and sales.

But machine learning models like linear regression unconsciously record some training data, and some training data involves people's privacy. It is inevitable that machine learning itself is vulnerable to security threats. In recent years, the security defense and privacy protection of machine learning have brought certain difficulties. The current researches on machine learning security defense and privacy protection are still in their infancy, and there are many problems to be solved, including the establishment of a sound evaluation mechanism. We should seek effective confrontation training methods and efficient encryption methods to protect user privacy. The most direct and effective way to protect privacy is to use encryption technology. However, current homomorphic encryption technology has too much computational overhead and cannot directly perform some non-polynomial operations in computer learning. User privacy is often protected at the expense of the accuracy of the target model. Therefore, researching efficient encryption methods to protect user privacy is an important research issue.

[1] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China.

[*] Corresponding Author: Danping Dong. Email: dongdp@139.com.

JBD.doi:10.32604/jbd.2019.08706

This paper proposes a Somewhat Homomorphic Encryption method to encrypt the data that needs to be trained and uses ciphertext data to train in the linear regression equation, while ensuring that the error between the training result and the original plaintext data's training result is in a controllable range. And after the training ciphertext is decrypted, it can be restored to the original plaintext training data without loss.

Below we will elaborate on this article in four aspects. The second part is related work and the third part is the application framework and specific process of ciphertext in linear regression equation. The fourth part is a summary of the work we have done.

2 Related work

2.1 Somewhat homomorphic encryption

Homomorphic encryption is a public-key cryptography that allows essential mathematical operations on data in the encrypted domain. As a homomorphic encryption, certainly, SHE (Somewhat Homomorphic Encryption) can satisfy the finite times of addition and a small number of multiplication operations on ciphertexts. It is based on a ring learning with errors (ring-LWE) homomorphic cryptosystem [Lindner and Peikert (2011)] and parametrized by the ring $R_q \triangleq \mathbb{Z}_q[x]/\langle x^n+1\rangle$. In this ring, the dimension n is a power of 2, an modulus prime number q and an error parameter σ that makes a definition of a discrete Gaussian error distribution $\chi = D_{\mathbb{Z}^n,\sigma}$ with standard deviation σ. According to a prime $t<q$, the message space of the scheme can be defined as $R_t = \mathbb{Z}_t[x]/\langle f(x)\rangle$. The goal of choosing these parameters (depending on the security parameter κ) in such a way is to guarantee correctness and security of SHE.

SH.Keygen (1^κ): Sample a ring element $s \overset{\$}{\leftarrow} \chi$ and set the secret key $sk \triangleq s$. Sample a uniformly random ring element $\mathrm{a}_1 \leftarrow \mathrm{R}_\mathrm{q}$ and an error $e \leftarrow \chi$. Meanwhile, compute the public key $pk \triangleq (a_0 = -(a_1 s + te), a_1)$.

Publish *pk* and keep *sk* secret.

SH.Enc (*pk, m*): Recall that our message space is R_t. Namely, we obtain a polynomial of degree n with coefficients in ℤt by encoding our message.

Given the public key $pk = (a_0, a_1)$ and a message $m \in R_q$, the encryption algorithm samples $u \leftarrow \chi$ and $f, g \leftarrow \chi$, and calculates the ciphertext

$$ct = (c_0, c_1) \triangleq (a_0 u + tg + m, a_1 u + tf) \tag{1}$$

SH.Dec ($sk, ct = (c_0, c_1, \dots, c_\delta$): To decrypt, we first calculate

$$m = \sum_{i=0}^{\delta} c_i s^i \in R_q \tag{2}$$

and obtain the message as $\widetilde{m}\ (mod\ t)$.

SH.Add (pk, ct_0, ct_1): Given two ciphertexts $ct = (c_0, c_1, \dots, c_\delta)$ and $ct' = (c_0', c_1', \dots, c_\gamma')$. Assume that $\delta = \gamma$, otherwise pad the shorter ciphertext with zeroes.

Through the simple component-wise addition of the ciphertexts, homomorphic addition can be finished. With this method, compute and output

$$ct_{add} = \left(c_0+c'_0, c_1+c'_1, \dots, c_{max(\delta,\gamma)}+c'_{max(\delta,\gamma)}\right) \in R_q^{max(\delta,\gamma)} \tag{3}$$

2.2 Simple linear regression equation

First, let's assume that we have a line

$$\hat{y} = \theta_1 x + \theta_2 \tag{4}$$

The Fig. 1 corresponding to this Eq. (4) is a straight line called the regression line. Where θ_1 is the slope of the regression line and θ_2 is the intercept of the regression. Then we need to determine if the line fits the points well. We can input a x to get the corresponding y value and then calculate the error between the two based on the real y value. The smaller the error, the better the straight line fit.

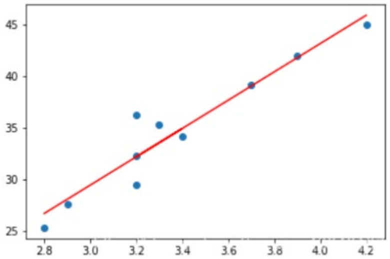

Figure 1: simple linear regression equation

Therefore, we can derive a loss function *y*-$\hat{y}$, the actual value minus the value calculated using the fit function is the error between the two. Even we can calculate the sum of all the errors, the smaller the value, the better the overall fit. The cost function is

$$J(\theta_1, \theta_2) = \frac{1}{2m}\sum_{i=0}^{m}(y_i - \hat{y})^2 \tag{5}$$

where *i* represents a sample in the range [*0, m*]. When the error function $J(\theta_1, \theta_2)$ is the smallest, the corresponding value of θ_1 and θ_2 is the optimal parameter. To get the optimal solution for θ_1 and θ_2, we use the gradient descent method to solve the linear regression equation.

$$\theta_1 = \theta_1 - \alpha\frac{\partial J}{\partial \theta_1} \tag{6}$$

$$\theta_2 = \theta_2 - \alpha\frac{\partial J}{\partial \theta_2} \tag{7}$$

where α is the learning rate. When the learning rate is too large, it does not converge, that is, it cannot find a global minimum or local minimum. If the learning rate is too small, it will waste a lot of time for calculation. And, the calculation method of the two partial derivatives is as follows

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m} x(y_i - \hat{y}) \tag{8}$$

$$\frac{\partial J}{\partial \theta_2} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}) \tag{9}$$

3 System framework

Generally, in the simple linear regression equation, the plaintext *x* is calculated by the linear regression equation to obtain the trained data $\hat{y}$. Now that the plaintext data *x* is encrypted into c_x, the encrypted data is put into the linear regression equation to train a data $\widehat{c_y}$. We hope to ensure that $\widehat{c_y}$ can still get the data trained in plaintext after decryption. The encryption process is represented by SHE encryption, the public key is *pk* and the private key is *sk*. The approximate flow chart is shown below.

The *x* and the predicted value $\hat{y}$ are encrypted with SHE, and encrypted to be represented as c_x， $c_{\hat{y}}$.

$$c_x = (c_0, c_1) \triangleq (a_0 u + tg + x, a_1 u + tf) \tag{10}$$

$$c_{\hat{y}} = (c_0, c_1) \triangleq (a_0 u + tg + \hat{y}, a_1 u + tf) \tag{11}$$

Therefore, the linear regression equation of the encryption domain can be expressed as

$$c_{\hat{y}} = \theta_1 c_x + \theta_2 \tag{12}$$

The error function of the corresponding ciphertext domain is

$$J(\theta_1, \theta_2) = \frac{1}{2m}\sum_{i=0}^{m}(c_y^i - \hat{c}_y) \tag{13}$$

The error function after partial derivation in the encryption domain is

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m} c_x(c_y^i - \hat{c}_y) \tag{14}$$

$$\frac{\partial J}{\partial \theta_2} = \frac{1}{m}\sum_{i=1}^{m}(c_y^i - \hat{c}_y) \tag{15}$$

In order to prove that the correct plaintext can still be obtained after decrypting the data in the ciphertext domain, we combine the partial derivative formula with the SHE encryption method to perform the following calculations.

$$\begin{aligned}\frac{\partial J}{\partial \theta_1} &= \frac{1}{m}\sum_{i=1}^{m}(a + x, b)[(a + y, b) - (a + \hat{y}, b)] \\ &\xRightarrow{SH.add} \frac{1}{m}\sum_{i=1}^{m}(a + x, b)\big[a + (y - \widehat{y}), b\big] \\ &\xRightarrow{SH.dec} \frac{1}{m}\sum_{i=1}^{m} x(y_i - \hat{y})\end{aligned} \tag{16}$$

$$
\begin{aligned}
&\frac{\partial J}{\partial \theta_2} = \frac{1}{m}\sum_{i=1}^{m}[(a + y, b) - (a + \hat{y}, b)] \\
&\xRightarrow{SH.add} \frac{1}{m}\sum_{i=1}^{m}\left[a + (y - \widehat{y)}, b\right] \qquad (17) \\
&\xRightarrow{SH.dec} \frac{1}{m}\sum_{i=1}^{m} x(y_i - \hat{y})
\end{aligned}
$$

where $a = a_0 u + tg$, $b = a_1 u + tf$. It can be proved by the above formula that the optimal solution can still be found in the ciphertext domain. In addition, the ciphertext can still recover the original plaintext data after the training.

4 Security analysis

As a kind of public key encryption, SHE consists of public key and private key. The public key *pk* is generated by a pseudo-random ring-LWE sample, and the private key *sk* is selected from the Gauss error distribution. If an attacker does not have a private key, he will usually attack it by replacing the public key pair *(u, v)*, but the result is still a random ring LWE sample. This ensures that the attacker cannot obtain the correct private key to protect the secret data. Therefore, she can provide a certain degree of security.

5 Conclusion and future work

This paper proposes a strategy for machine learning in the encryption domain. It is mainly for the comparison of the basic one-dimensional linear regression equation. It is proved that the machine learning in the ciphertext domain is feasible by the algorithm in the paper, and the ciphertext data after the machine training can also recover the original plaintext. This provides some protection for improving the safety of machine learning. Since multiple linear regression equations have important significance in the field of data prediction, we will further study the encryption domain of multiple linear regression equations in the future to ensure a more reliable role in protecting data security.

References

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A. (2017): Practical secure aggregation for privacy-preserving machine learning. *ACM Sigsac Conference on Computer & Communications Security*, pp. 1175-1191.

Li, P.; Li, J.; Huang, Z. G.; Li, T.; Gao, C. Z. et al. (2017): Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, vol. 74, no. C, pp. 76-85.

Lindner, R.; Peikert, C. (2011): Better key sizes (and attacks) for LWE-based encryption. *International Conference on Topics in Cryptology: Ct-Rsa*, pp. 1175-1191.

Liu, M.; Jiang, H.; Chen, J.; Badokhon, A. (2017): A collaborative privacy-preserving deep learning system in distributed mobile environment. *International Conference on Computational Science & Computational Intelligence.*

Phong, L. P.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. (2017) Privacy-preserving deep learning: revisited and enhanced. *International Conference on Applications and Techniques in Information Security*, pp.100-110.

Shokri, R.; Shmatikov, V. (2015): Privacy-preserving deep learning. *Allerton Conference on Communication*, pp. 1310-1321.

Zhang, Q.; Yang, L. T.; Chen, Z. (2016): Privacy preserving deep computation model on cloud for big data feature learning. *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1351-1362.