Privacy-Preserving Method for Neural Network Training

Can Tu, Yuchen Ding*

Shanghai Jiaotong University

January 11, 2019

I. Introduction

With the development of cloud-computing device, in practice multiple parties may collaborate through neural network training on the union of their respective data sets. Thus, it is more convenient for users across the Internet to conduct collaborative learning.

In such distributed environment, the protection of data privacy for each participant becomes a big issue. In many situations like security department and medical researches, the privacy of the original data must be protected according to some privacy rules. So in pursuit of embracing the joint learning, it is imperative to provide a solution that allow users to conduct collaborative learning without exposure of their private data.

Challenges. In order to provide practical solutions for privacy-preserving neural network learning, three main challenges need to be met simultaneously: 1) Secure computation of various operations, e.g. addition, scalar product and the nonlinear sigmoid function, which are needed by the network algorithm; 2) To ensure the practicality of the proposed solution, the computation/communication cost introduced to each participant shall be affordable; 3) For collaborative training, the training data sets may be owned by different parties and partitioned in arbitrary ways rather than a single way of partition.

In our proposed method, we focus on multilayer neural network. In some referenced work, gradient-descent methods are used to ensure privacy-perservation. But for multilayer neural network model, gradient-descent methods are elegant in its generality and restricted in practice for such simple model. Our method is based on two-party distributed algorithm for privacy-preserving backpropagation training.

*Group3

II. Background and Related Works

Few privacy-preserving neural network learning schemes have been proposed, which are used for our reference and we benefit a lot from them. The notion of privacy-preserving data mining problem was proposed in 2000. Two different papers solved the problem that privacy constraints on distributed environment. Agrawal et al. proposed randomization to preserve sensitive data, in other words add noise to original data, at the cost of accuracy. Lindell and Pinkas introduced cryptographic tools for higher accuracy but the computation complexity is respectively giant when data becomes larger. And after the two papers, a lot of work have been done in data mining with privacy constraints such as clustering, classification.

Constraints for existing solutions in data mining is that most of them are randomization-based approches, which result in the privacy generally was limited to some extent. Cryptography-based approaches provide better accuracy and privacy guarantee compared to randomization-based schemes. But the computation process can be tarnally

pursuming.

For privacy preserving neural network learning, few work have been done. Barni et al proposed security algorithms for three scenarios in neural networks. Chen and Zhong [6] propose privacy preserving back-propagation neural network learning algorithm when training data is vertically partitioned. Their algorithm provides strong privacy guaranty to the participants. The solution when the training data is horizontally partitioned data is much easier since all the data holders can train the neural network in turns.

III. Motivation

Our motivation comes from the limitations of accuracy and effeciency for existing works. In order to improve the computation rate and promising accuracy, we use a three-layer network as our model, so when the data set become larger, our training time can be guaranteed.

IV. Problem Formulation

i. Semihonest Model

As many existing reference paper we have found, we use semihonest model in our method. Semihonest model is a standard adversary model in cryptography.

When computing function in a distributed fashion, semihonest model requires that each party that participates in the computation follow the algorithm, but a party may try to learn additional information by analyzing the messages that she receives during the execution. In order to guarantee the security of distributed algorithm of computing , it must be ensured that each party can learn nothing beyond what can be implied by its own input and output.

Based on semihonest model, our problem of privacy-preserving neural network learning is stated below.

ii. Privacy-Preserving Two-Party Distributed Neural Network Learning

Suppose that a set of training samples are vertically partitioned between two parties A and B. A holds a data set D_1 with m_A attributes for each data entry. B holds a data set D_2 with m_B attributes for each data entry. We denote one data entry in D_1 as x_A=(x_1,x_2,...,x_{mA}) and in D_2 x_A=(x_{mA+1},...,x_{mA+mB}).

Privacy-preserving two-party distributed neural network training is that, in each round of neural network learning, two parties jointly compute the additive values of connection weights without compromising their privacy of input data. Formally, with training samples x_A and x_B from party A and B, respectively, and a target value t(x), our goal is to let each party get her own share of the additive value of each weight Δw, without revealing the original training data x_A or x_B to each other.

V. Proposed Methods

Firstly we present a privacy-preserving distriuted algorithm for training the neural networks with backpropagation algorithm.

Different steps and details will be introduced later.

i. Privacy-Preserving Neural Network Training Algorithm

As we mentioned in the preamble, we use a neural network of three layers simply, where the hidden-layer activation function is sigmoid and the output-layer is linear.

Back-Propagation neural network learning algorithm is mainly composed of two stages: feed forward and error backpropagation. Given a neural network with a-b-c configuration, one input vector is denoted as (x_1,x_2,...,x_a). The values of hidden-layer nodes are denoted as (h_1,h_2,...,h_b), and the values of output-layer nodes are (o_1,o_2,...,o_c). $w^h{}_{jk}$ denotes the weight connecting the input-layer node k and the hidden-layer node j. $w^o{}_{ij}$ denotes the weight connecting j and the output-

layer node i, where $1 \leq k \leq a$, $1 \leq j \leq b$, $1 \leq i \leq c$.

We use mean square error(MSE) as the error function in the backpropagation algorithm $e = 1/2 \sum_i (t_i - o_i)^2$.

$$\Delta w_{ij}^o = -(t_i - o_i) h_j \quad (1)$$

$$\Delta w_{jk}^h = -h_j(1 - h_j) x_k \sum_{i=1}^{c} [(t_i - o_i) * w_{ij}^o)] \quad (2)$$

For each training iteration,the input of the privacy-preserving backpropagation training algorithm is (x_A,x_B,t(x)), where x_A is held by party A, while x_B is held by party B. t(x) is the target vector of the current training data and it is known to both parties. The output of algorithm is the connection weights of output layer and hidden layer.

The main idea of the algorithm is to secure each step in the nonprivacy-preserving backpropagation algorithm, with two stages, feedforward and backpropagation. In each step, neither the input data from the other party nor the intermediate results can be revealed. Morever, we use the piecewise linear approximation to compute the sigmoid function, which will be introduced later.

To hide the intermediate results such as the values of hidden-layer nodes, the two parties randomly share each result so that neither of the two parities can imply the original data information from the intermediate results. Here by randomly share, we mean that each party holds a random number and the sum of the two random numbers equals to the intermediate result. Note that with intermediate results randomly shared among two parties, the learning process can still securely carry on to produce the correct learning result.

After the entire process of private training, without revealing any raw data to each other, the two parties jointly establish a neural network representing the properties of the union data set.

Algorithm 1: Privacy-Preserving Distributed Algorithm for Backpropagation Training

Initialize all weights to small random numbers, and make them known to both parties.
For all trainint sample do
Step 1: feedforward stage

1. For each hidden layer node h_j, party A computes $\sum_{k<m_A}(w_{jk}^h)x_k$, and party B computes $\sum_{m_A<k<m_A+m_B}(w_{jk}^h)x_k$.
2. Using piecewise linear approximation, parties A and B jointly compute the sigmoid function for each hidden-layer node h_j, obtaining the random shares h_{j1} and h_{j2}. $h_{j1} + h_{j2} = f(\sum_k (w_{jk}^h)x_k)$.
3. For each output layer node o_i, party A computes $o_{i1} = \sum_i (w_{ij}^o)h_{j1}$ and party B computes $o_{i2} = \sum_i (w_{ij}^o)h_{j2}$. $o_i = o_{i1}+o_{i2} = \sum_i (w_{ij}^o)h_{j1}+\sum_i (w_{ij}^o)h_{j2}$.

Step 2: backpropagation stage

1. For each output layer weight w_{ij}^o, parties A and B securely compute the product $h_{j1}o_{i2}$, obtaining random shares $r_1 1$ and $r_1 2$. Similarly, they compute the random partitions of $h_{j2}o_{i1}$, $r_2 1$, $r_2 2$. Parrt A computes $\Delta_1 w_{ij}^o = (o_{i1}\text{-}t_i)h_{j1} + r_{11} + r_{21}$ and Party B computes $\Delta_2 w_{ij}^o = (o_{i2}\text{-}t_i)h_{j2} + r_{12} + r_{22}$.
2. For each hidden layer weight w_{jk}^h, parties A and B jointly compute random shares q_1 and q_2. If $k \leq m_A$, get $x_k q_2 = r_{61}+r_{62}$. Then, $\Delta_1 w_{jk}^h = q_1 x_k + r_{61}$ and $\Delta_2 w_{ij}^o = r_{62}$. If $m_A < x_k \leq m_A + m_B$, $\Delta_1 w_{jk}^h = r_6 1$ and $\Delta_2 w_{ij}^o = q_2 x_k + r_{62}$.
3. A (Similarly for B) sends $\Delta_1 w_{ij}^o$ and $\Delta_1 w_{jk}^h$ to B. Then A and B compute for each hidden-layer weight and each output-layer weight.

So using Algorithm 1, the hidden-layer weights can be updated correctly by two parties without compromising their data privacy.

ii. Securely Computing the Linear Sigmoid Function

Cryptographic tools work in finite fields and thus cannot be directly applied to the secure computation of functions such as sigmoid. Approximating the activation function in a piecewise way offers us an opportunity to apply cryptographic tools to make the computation of sigmoid function privacy preserving.

The following equation is a piecewise linear approximation of the sigmoid function $1/1+e^{-}x$.

$$y(x) = \begin{cases} 1 & x > 8 \\ 0.015625x + 0.875 & 4 < x \leq 8 \\ 0.03125x + 0.8125 & 2 < x \leq 4 \\ 0.125x + 0.625 & 1 < x \leq 2 \\ 0.25x + 0.5 & -1 < x \leq 1 \\ 0.125x + 0.375 & -2 < x \leq -1 \\ 0.03125x + 0.1875 & -4 < x \leq -2 \\ 0.015625x + 0.125 & -8 < x \leq -4 \\ 0 & x \leq -8 \end{cases} \tag{1}$$

Our secure distributed algorithm is based on the piecewise linear approximation sigmoid function.

Algorithm 2: Securely Computing the Piecewise Linear Sigmoid Function

Step 1:
Party A generates a random number R and computes $y(x_1+i)$-R for each i. Define m_i = $y(x_1+i)$-R. Party A encrypts each m_i using random noise and gets $E(m_i,r_i)$.

Step 2:

Party B picks $E(m_{x_2},r_{x_2})$. Rerandomize it and sends $E(m_{x_2},r')$ back to A.

Step 3:

Party A partially decrypts $E(m_{x_2},r')$ and sends the partially decrypted message to B.

Step 4:

Party B finally decrypts the message to get m_{x_2} = $y(x_1+x_2)$-R. Note R is noly known to A and m_2 is only known to B.

Although each party only holds one part of the input to the sigmoid function, this algorithm enables them to compute the approximate value of the function without knowing the part of input from the other party. Actually, in this algorithm, there is no way for each party to explore the input of the other, but the function value can still be computed.

iii. Privacy-Preserving method for Computing Product

To securely compute product, some existing secure multi-party computation protocols for dot product can be utilized.

Algorithm 3: Securely computing the scalar Product

Party A with Private vectors **x** and B with **y**
for i=1 to N
  A generates random number r_i
  A send $c_i = E_p k(x_i, r_i)$ to B
B computes $w = \prod_{i=1}^{N} c_i^{y_i}$
B sends w to A A computes $z = D_s k(w) = \mathbf{x} \cdot \mathbf{y}$

Before applying Algorithm 3, a preprocessing step is needed to convert the input of each party into an asymmetric encrypted text.

VI. Experiments

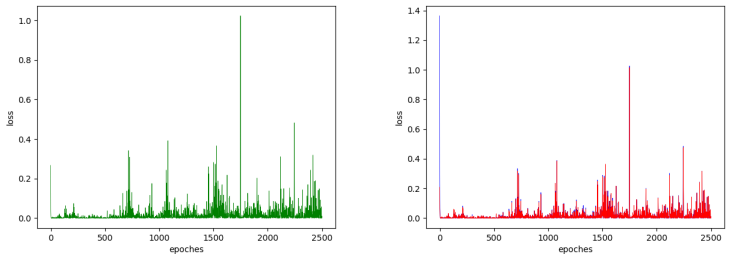
To illustrate the method we use to ensure the security between networks during training, we establish a typical model, 2-party collaborative training of BPN (Back-propagation network). As a comparison, we add a test of normal single network finishing the same task. And after that, we let 2 networks communicate without leaking their private data (including input vector and weights).

The algorithms are implemented in Pytorch and complied with Python3.7. The experiments were executed on a virtual machine with Linux 16.04 OS.

The basic setup is classical. We utilize 2 famous datasets in this work, Maruti-data from stock exchange society and Real-Estate-Database from a survey on people's accomodation and fortune. Input is divided into 2 parts as private data for networks and they cannot know other's input since. Model is 3-layer linear-sigmoid-linear network. Activation functions are flexible as we can simulate the functions when convenient.

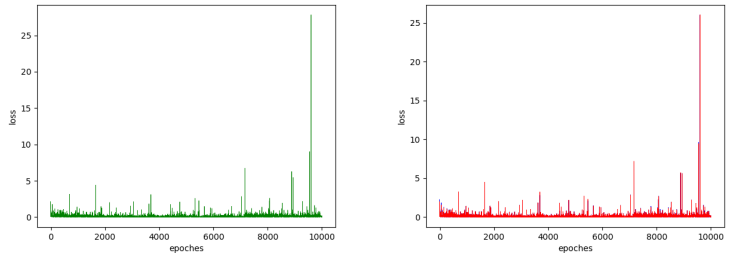
The result figures are displayed below.

Privacy is protected obviously during process above. While we need to communicate (exchange data message) in order to



(a) the loss during time of single complete network (b) the loss during time of 2-party collaborative networks

Figure 1: *Tests on Maruti dataset*



(a) the loss during time of single complete network (b) the loss during time of 2-party collaborative networks

Figure 2: *Tests on Estate dataset*

learn together just like a single (complete) network. This comunication works through a encryption-decryption message transfer process, in which two party encode the data only belong to it into cyphertext and send to the other and get a couterpart meanwhile.

The results of loss function figure have shown that this method also keeps the availability and efficiency of whole training work. Thus we can primitively proove the basic ideas of our work, according to the plan.

VII. Conclusion

In this project, we present a efficient but accuracy-promising privacy-preserving method for backpropagation neural network learning. The algorithm guarantees privacy in a standard cryptographic model, the semihonest model.

The drawback of our work is that we only considered backpropagation neural network, and other types of training is to be extended for us.