International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-4, July 2015
ISSN: 2395-3470
www.ijseas.com

PRIVACY PRESERVING BACK-PROPAGATION NEURAL NETWORK IN CLOUD COMPUTING

Gunda Satish Kumar[1], Mr.K.Nagaraju[2], Dr.Y.Venkateswarulu[3]

[1]Mtech Student, CSE, Giet Engineering College, Rajahmundry, A,P, India
[2]Assoc Professor, Dept of CSE, Giet Engineering College, Rajahmundry, A,P, India
[3]Professor and HOD Dept of CSE, Giet Engineering College, Rajahmundry, A,P, India

ABSTRACT

To improve the accuracy of learning result, in practice multiple parties may collaborate through conducting joint Back-Propagation neural network learning on the union of their respective data sets. We find the solutions of Privacy Preserving Back Propagation Algorithm for a Vertically Partitioned Dataset. To improve the learning, Enhanced data is more important to find the exact privacy concern of each data holder by extending the privacy preservation suggested to original learning algorithms. In this paper, we try to improve preserving the privacy in an important multilayer neural networks and learning model. We present a privacy preserving multiparty distributed algorithm of back propagation which allows a neural network to be trained without requiring either party to reveal her data to the others.

Indexed terms: Back-propagation, Privacy Preserving, Neural Network

INTRODUCTION:

Back-propagation [1] is an effective method for learning neural networks and has been widely used in various applications. As compared the volume of quality data to learning with only local data set, combined learning algorithm that improves the learning accuracy by incorporating more data sets into the learning process the participating data set address the problem not only on their own data sets, but also on others data sets. As compared to learning with only local data set, collaborative learning improves the learning accuracy by incorporating more data sets into the learning process [2], [3], [4].

Back-Propagation Neural Network Learning

Back-Propagation neural network learning algorithm is mainly composed of two stages:

1. Feed forward

ERROR BACK - PROPAGATION

In the Feed Forward Stage, values at each layer are calculated using the weights, the sigmoid unction, and the values at the previous layer. In the Back - Propagation stage, the algorithm checks whether the error between output values and target values is within the threshold.

Back-propagation is an effective method for learning neural networks and has been widely used in various applications. The accuracy of the learning result, despite other facts, is highly affected by the volume of high quality data used for learning. As compared to learning with only local data set, collaborative learning improves the learning accuracy by incorporating more data sets into the learning process the participating parties carry out learning not only on their own data sets, but also on others data sets. With the recent remarkable growth of new computing infrastructures such as Cloud Computing, it has been more convenient than ever for users across the Internet, some of us don't know each other they to conduct joint/collaborative learning through the shared infrastructure of cloud computing.

In order to use the Internet wide collaborative learning, it is imperative to provide a solution that allows the participants to correct the values, who lack mutual trust, to conduct neural network learning jointly without disclosing their respective private data sets. Preferably, the solution shall be efficient and scalable enough to support an arbitrary number of participants, each possessing arbitrarily partitioned data sets.

Neural Network

Neural networks have seen an explosion of interest over the last few years and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics.

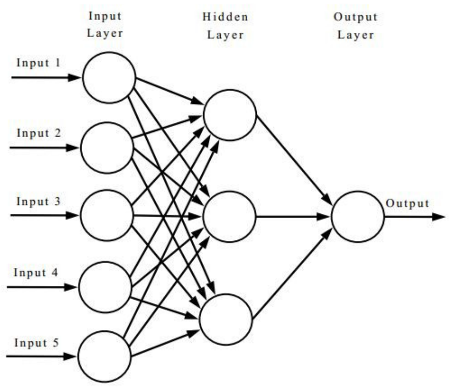


Figure 1: **Neural Network**

Proposed MThod:

Privacy Preserving Neural Network Learning Partitioned Data

Barni, M., Orlandi, C., &Piva, A. (2006)[6] Presents a new approach for privacy preserving neural network training. Several studies have been devoted to privacy preserving supervised model learning, but little work has been done to extend neural network learning with a privacy preserving protocol. Neural networks are popular for many applications, among else those calling for a robust learning algorithm. In this study, we elaborate on privacy preserving classification as well as regression with neural networks on horizontally partitioned data. We consider a scenario of more than two parties that are semi-honest but curious. We extend the neural network classification algorithm with protocols for secure sum and secure matrix addition. The extended algorithm does not fully guarantee privacy in the sense, but we show that the information revealed is not associated to a specific party and could also be derived by juxtaposing the local data and the final model.

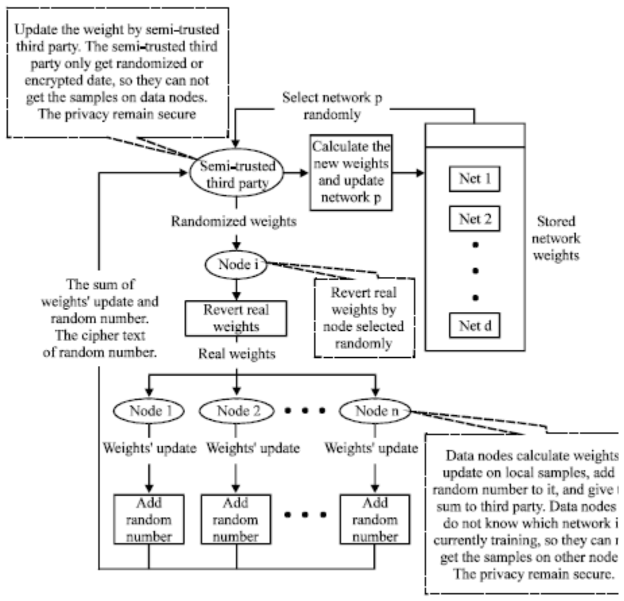

Update the weight by semi-trusted third party. The semi-trusted third party only get randomized or encrypted date, so they can not get the samples on data nodes. The privacy remain secure
Select network p randomly
Semi-trusted third party
Calculate the new weights and update network p
Net 1
Net 2
Net d
Stored network weights
Randomized weights
Node i
The sum of weights' update and random number. The cipher text of random number.
Revert real weights
Real weights
Revert real weights by node selected randomly
Node 1
Node 2
Node n
Weights' update
Weights' update
Weights' update
Add random number
Add random number
Add random number
Data nodes calculate weights update on local samples, add random number to it, and give sum to third party. Data nodes do not know which network is currently training, so they can get the samples on other node The privacy remain secure.

Fig 2: **Neural Network on Horizontally portioned Data**

Neural Networks are nature inspired computation models that are widely used for regression and classification tasks. A neural network consists of nodes and weighted edges. In general, we distinguish feed forward and recurrent neural networks. In a feed forward network, the information is transmitted only in one direction, from the input nodes, through the hidden nodes (if any) to the output nodes. A feed forward network has therefore no cycles or loops. In recurrent neural networks, connections between nodes form a directed cycle.

Neural Networks have been an active research area for decades. However, Ankur Bansal Tingting Chen Sheng Zhong (2010) [8] are privacy bothers many when the training dataset for the neural networks is distributed between two parties, which is quite common nowadays. Existing cryptographic approaches such as secure scalar product protocol provide a secure way for neural network learning when the training dataset is vertically partitioned. In this paper we present a privacy preserving algorithm for the neural network learning when the dataset is arbitrarily partitioned between the two parties. We show that our algorithm is very secure and leaks no knowledge (except the final weights learned by both parties) about other party's data. We demonstrate the efficiency of our algorithm by experiments on real world data. Privacy preserving neural network learning With the invention of new technologies, whether it is data mining, in databases or in any networks, resolving privacy problems has become very important. Because all sorts of data is collected from many sources, the field of machine learning is equally growing and so are the concerns regarding the privacy. Data providers for machine learning are not willing to train the neural network with their data at the expense of privacy and even if they do participate in the training they might either remove some information from their data or can provide false information.

Problem Statement: In this paper, we intend at enabling numerous parties to together conduct BPN network learning without enlightening their personal data. The input data sets owned by the parties can be arbitrarily partitioned. The computational and communicational costs on each party shall be practically efficient and the system shall be scalable.

Specifically, we consider a 3-layer (a-b-c configuration) neural network for simplicity but it can be easily extended to multilayer neural networks. The learning data set for the neural network, which has N samples (denoted as vector $\{x_1^m, x_2^m, \ldots, x_a^m\}$, $1 \leq m \leq N$ is arbitrary partitionedinto $Z(Z \geq 2)$ subsets. Each party P_s holds x_{1s}^m, x_{2s}^m, $\ldots, x_{as}^m$ and has

$$x_{11}^m + x_{12}^m + \ldots + x_{1a}^m = x_1^m$$

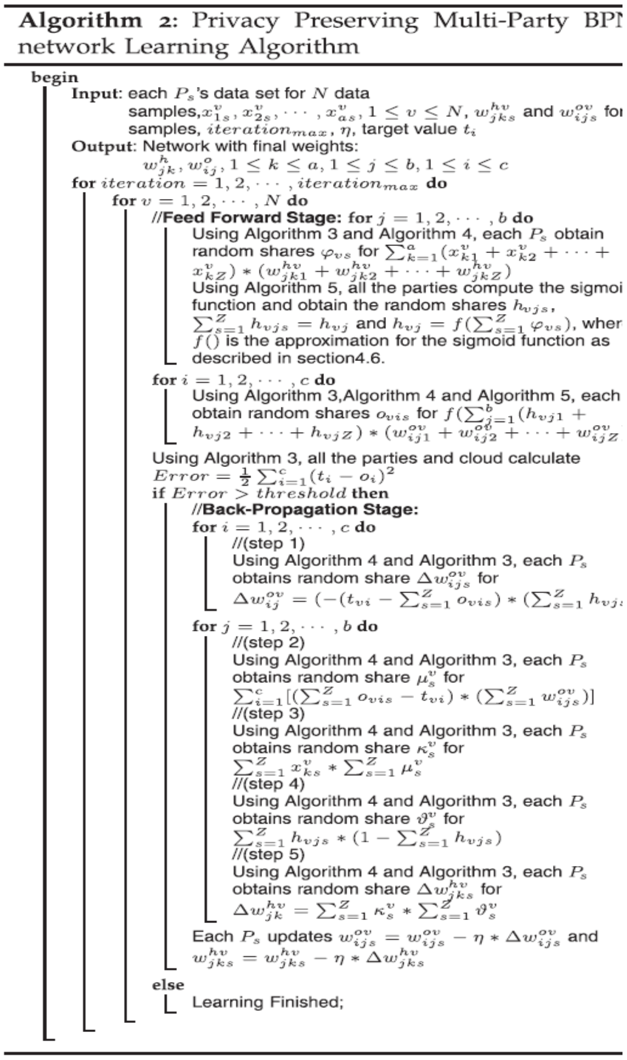
$x_{a1}^{m} + x_{a2}^{m} + \ldots + x_{aa}^{m} = x_{a}^{m}$

For collaborative learning, the main task for all the parties is to jointly execute the operations defined in the Feed Forward stage and the Back-Propagation stage as shown in Algorithm 1. During each learning stage, except for the final learned network, neither the input data of each party nor the intermediate results (i.e., weights, value of hidden layer node, value of output layer node) generated can be revealed to anybody other than TA.

To achieve the above goals, the main idea of our proposed scheme is to implement a privacy preserving equivalence for each step of the original BPN network learning algorithm described in Algorithm 1. Different from the original BPN network learning algorithm, our proposed scheme lets each party encrypt her/ his input data set and upload the encrypted data to the cloud, allowing the cloud servers to perform most of the operations, i.e., additions and scalar products. To support these operations over ciphertexts, we adopt and tailor the BGN "doubly homomorphic" encryption [7] for data encryption. Nevertheless, as the BGN algorithm just supports one step multiplication over ciphertext, the intermediate results, for example, the intermediate products or scalar products, shall be first securely decrypted and then encrypted to support consecutive multiplication operations as described in Algorithm 1. For privacy preservation, however, the decrypted results known to each party cannot be the actual intermediate values, for example, the values of the hidden layer. For this purpose, design a secret sharing algorithm that allows the parties to decrypt only the random shares of the intermediate values. The random shares allow the parties to collaboratively execute the following steps without knowing the actual intermediate values. Data privacy is thus well protected. The overall algorithm is described in Algorithm 2, which is the privacy preserving equivalence of Algorithm 1.

Algorithm 1: Back-Propagation Neural Network Learning Algorithm

```
Input: N input sample vectors $\vec{V}_i, 1 \le i \le N$, with $a$ dimensions,
       $iteration_{max}$, learning rate $\eta$, target value $t_i$, sigmoid function
       $f(x) = \frac{1}{1+e^{-x}}$
Output: Network with final weights:
       $w_{jk}^{h}, w_{ij}^{o}, 1 \le k \le a, 1 \le j \le b, 1 \le i \le c$
begin
    Randomly Initialize all $w_{jk}^{h}, w_{ij}^{o}$.
    for $iteration = 1, 2 \cdots, iteration_{max}$ do
        for $sample = 1, 2 \cdots, N$ do
            //Feed Forward Stage:
            for $j = 1, 2 \cdots, b$ do
                $h_j = f(\sum_{k=1}^{a} x_k * w_{jk}^{h})$
            for $i = 1, 2 \cdots, c$ do
                $o_i = f(\sum_{j=1}^{a} h_j * w_{ij}^{o})$
            if $Error = \frac{1}{2}\sum_{i=1}^{c}(t_i - o_i)^2 > threshold$ then
                //Back-Propagation Stage:
                $\Delta w_{ij}^{o} = -(t_i - o_i) * h_j$
                $\Delta w_{jk}^{h} = -h_j(1 - h_j)x_k \sum_{i=1}^{c}[(t_i - o_i) * w_{ij}^{o})]$
                $w_{ij} = w_{ij} - \eta \Delta w_{ij}$
                $w_{jk}^{h} = w_{jk}^{h} - \eta \Delta w_{jk}^{h}$
            else
                //Learning Finish
                break
```



Algorithm 2: Privacy Preserving Multi-Party BP
network Learning Algorithm

```
begin
    Input: each $P_s$'s data set for $N$ data
           samples,$x^v_{1s}, x^v_{2s}, \cdots, x^v_{as}, 1 \le v \le N$, $w^{hv}_{jks}$ and $w^{ov}_{ijs}$ fo
           samples, $iteration_{max}$, $\eta$, target value $t_i$
    Output: Network with final weights:
            $w^h_{jk}, w^o_{ij}, 1 \le k \le a, 1 \le j \le b, 1 \le i \le c$
    for $iteration = 1, 2, \cdots, iteration_{max}$ do
        for $v = 1, 2, \cdots, N$ do
            //Feed Forward Stage: for $j = 1, 2, \cdots, b$ do
                Using Algorithm 3 and Algorithm 4, each $P_s$ obtain
                random shares $\varphi_{vs}$ for $\sum_{k=1}^{a}(x^v_{k1} + x^v_{k2} + \cdots +$
                $x^v_{kZ}) * (w^{hv}_{jk1} + w^{hv}_{jk2} + \cdots + w^{hv}_{jkZ})$
                Using Algorithm 5, all the parties compute the sigmoi
                function and obtain the random shares $h_{vjs}$,
                $\sum_{s=1}^{Z} h_{vjs} = h_{vj}$ and $h_{vj} = f(\sum_{s=1}^{Z} \varphi_{vs})$, wher
                $f()$ is the approximation for the sigmoid function as
                described in section4.6.
            for $i = 1, 2, \cdots, c$ do
                Using Algorithm 3,Algorithm 4 and Algorithm 5, each
                obtain random shares $o_{vis}$ for $f(\sum_{j=1}^{b}(h_{vj1} +$
                $h_{vj2} + \cdots + h_{vjZ}) * (w^{ov}_{ij1} + w^{ov}_{ij2} + \cdots + w^{ov}_{ijZ}$
            Using Algorithm 3, all the parties and cloud calculate
            $Error = \frac{1}{2}\sum_{i=1}^{c}(t_i - o_i)^2$
            if $Error > threshold$ then
                //Back-Propagation Stage:
                for $i = 1, 2, \cdots, c$ do
                    //(step 1)
                    Using Algorithm 4 and Algorithm 3, each $P_s$
                    obtains random share $\Delta w^{ov}_{ijs}$ for
                    $\Delta w^{ov}_{ij} = (-(t_{vi} - \sum_{s=1}^{Z} o_{vis}) * (\sum_{s=1}^{Z} h_{vj}$
                for $j = 1, 2, \cdots, b$ do
                    //(step 2)
                    Using Algorithm 4 and Algorithm 3, each $P_s$
                    obtains random share $\mu^v_s$ for
                    $\sum_{i=1}^{c}[(\sum_{s=1}^{Z} o_{vis} - t_{vi}) * (\sum_{s=1}^{Z} w^{ov}_{ijs})]$
                    //(step 3)
                    Using Algorithm 4 and Algorithm 3, each $P_s$
                    obtains random share $\kappa^v_s$ for
                    $\sum_{s=1}^{Z} x^v_{ks} * \sum_{s=1}^{Z} \mu^v_s$
                    //(step 4)
                    Using Algorithm 4 and Algorithm 3, each $P_s$
                    obtains random share $\vartheta^v_s$ for
                    $\sum_{s=1}^{Z} h_{vjs} * (1 - \sum_{s=1}^{Z} h_{vjs})$
                    //(step 5)
                    Using Algorithm 4 and Algorithm 3, each $P_s$
                    obtains random share $\Delta w^{hv}_{jks}$ for
                    $\Delta w^{hv}_{jk} = \sum_{s=1}^{Z} \kappa^v_s * \sum_{s=1}^{Z} \vartheta^v_s$
                Each $P_s$ updates $w^{ov}_{ijs} = w^{ov}_{ijs} - \eta * \Delta w^{ov}_{ijs}$ and
                $w^{hv}_{jks} = w^{hv}_{jks} - \eta * \Delta w^{hv}_{jks}$
            else
                Learning Finished;
```

PERFORMANCE EVALUATION

To measure the impact of our improvements from existing schemes and evaluate the practicality of our privacypreserving BPN network learning scheme, we numerically analyze it and fully implemented it on Amazon EC2 cloud.

NUMERICALLY ANALYSIS

In this section, we numerically evaluate the performance of our proposed scheme in terms of computation cost and communication cost and compare it with the existing techniques. For expression simplicity, in the following part of this section, we denote time cost of one multiplication operation on Group G as MUL1 and that of one exponentiations operation on Group G as EXP. For the neural network configuration (a-b-c) (), first each party Ps needs to encrypt all its privacy data once using Algorithm 3 with 2ðns þ b þ cÞ EXP and ðns þ b þ cÞ MUL, where ns is the number of data attributes held by Ps, a, b, and c represent the number of input layer nodes, hidden layer nodes, and output layer nodes, respectively. Note that this is the one-time cost performed before the learning process startsDataset Preparation. Before outsourcing his/her privatedata to cloud for BPN network learning, every party Psneeds to encrypt the data with Algorithm 3. Fig. 2 shows thatthe data encryption time for each party is greatly affected bythe number of samples and features in the data set, whichvaries from 4.12 to 120.21 s for different data sets. However,this is a one-time cost and can be pre-computed offline.Collaborative Learning. For multiparty privacy-preservingBPN network learning with fixed neural network architecture,there are two factors—the number of party and thesize of data set—that mainly affect the system performanceaccording to the experimental results of existing schemes[6]. As the number of participants increases the operationsfor each party to share intermediate results and decryptthat the final learning result remain the same in ourproposed scheme. As shown in Fig. 3a, when the number of parties varies

from 2 to 8, the overall learning time stays stable in our scheme, which are about 10.68, 21.86, and 36.69 minutes for Iris, Diabetes, and kr-vs-kp, respectively. The learning time does not change with the number of parties because the cloud performs most learning operations in parallel without learning the private data. However, as demonstrated in Fig. 3b, [5], when extended to the multiparty setting, introduces a quadratically increasing cost in the number of parties for each Ps, which can be up to 961.52, 2,152.12, and 5,750.08 minutes for Iris, Diabetes, and kr-vs-kp, respectively, and makes the learning scheme less practical. Notably, even in the two-party setting, for which [6] was designed, our scheme can still save at least 37 percent learning time as shown in Fig. 4.

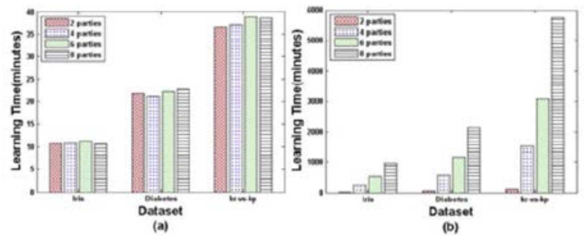


Fig. 3. Learning time for different party number and data set:Learning time of our scheme; (b) learning time of Chen's scheme

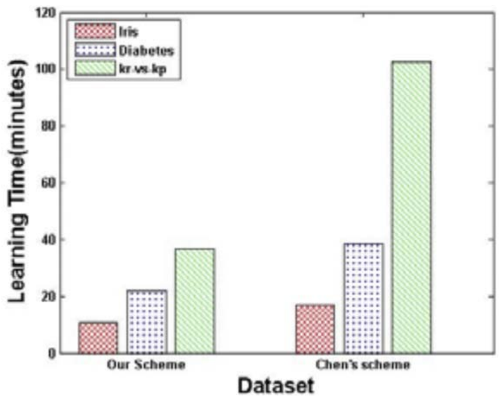


Fig. 4. Two-party scenario learning efficiency comparison

CONCLUSION

In this work, we proposed the first secure and practicalmultiparty BPN network learning scheme over arbitrarilypartitioned data. In our proposed approach, the partiesencrypt their arbitrarily partitioned data and upload thecipher texts to the cloud. The cloud can execute mostoperations pertaining to the BPN network learning algorithmwithout knowing any private information. The cost ofeach party in our scheme is independent to the number ofparties. This work tailors the BGN homomorphic encryptionalgorithm to support the multiparty scenario, whichcan be used as an independent solution for other relatedapplications. Complexity and security analysis shows that our proposed scheme is scalable, efficient, and secure. Oneinteresting future work is to enable multiparty collaborativelearning without the help of TA.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318-362, MIT Press, 1986.

[2] S. Stolfo, A.L.P.S. Tselepis, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan, and P.K. Chan, "JAM: Java Agents for Meta-Learning over Distributed Databases," Proc. Third Int'l Conf. Knowledge Discovery and Data Mining, pp. 74-81, 1997.

[3] K. Flouri, B. Beferull-lozano, and P. Tsakalides, "Training a SVMBased Classifier in Distributed Sensor Networks," Proc. 14th European Signal Processing Conf., pp. 1-5, 2006.

[4] B. Yang, Y.-d. Wang, and X.-h. Su, "Research and Design of Distributed Neural Networks with Chip Training Algorithm," Proc. First Int'l Conf. Advances in Natural

Computation (ICNC ’05)- Vol. Part I, pp. 213-216, 2005.

[5] T. Chen and S. Zhong, “Privacy-Preserving Backpropagation Neural Network Learning,” IEEE Trans. Neural Network, vol. 20, no. 10, pp. 1554-1564, Oct. 2009.

[6] Barni. M., Orlandi, C., &Piva, A. (2006). A Privacy-Preserving Protocol for Neural-Network-Based Computation Trans

[7] L. Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, “Handwritten Digit Recognition with a Back-Propagation Network,” Proc. Advances in Neural Information Processing Systems, pp. 396-404, 1990.

[8] S.D.C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Over-Encryption: Management of Access Control Evolution on Outsourced Data,” Proc. 33rd Int’l Conf. Very Large Data Bases (VLDB ’07), pp. 123-134, 2007.