Privacy-Preserving Protocols for Perceptron Learning Algorithm in Neural Networks

Saeed Samet  
School of Information Technology and Engineering (SITE)  
University of Ottawa  
Ottawa, Canada K1N 6N5  
Email: saeed.samet@uottawa.ca

Ali Miri  
School of Information Technology and Engineering (SITE)  
University of Ottawa  
Ottawa, Canada K1N 6N5  
Email: ali.miri@uottawa.ca

Abstract—Neural networks have become increasingly important in areas such as medical diagnosis, bio-informatics, intrusion detection, and homeland security. In most of these applications, one major issue is preserving privacy of individual’s private information and sensitive data. In this paper, we propose two secure protocols for perceptron learning algorithm when input data is horizontally and vertically partitioned among the parties. These protocols can be applied in both linearly separable and non-separable datasets, while not only data belonging to each party remains private, but the final learning model is also securely shared among those parties. Parties can jointly and securely apply the constructed model to predict the output corresponding to their target data. Also, these protocols can be used incrementally, i.e., they process new coming data, adjusting the previously constructed network.

Index Terms—Security and Privacy Preserving; Neural Networks; Data Mining and Machine Learning; Distributed Data Structures.

I. INTRODUCTION

Preserving the privacy of sensitive data in machine learning and data mining methods is an important issue in data communication and knowledge base systems. Therefore, many protocols have been proposed for different methods such as classification using decision tree [1], [2], [3], [4], [5], [6], [7], [8], association rule mining [9], [10], [11], [12], and clustering [13], [14], [15], [16], [17]. The common purpose in these protocols is to keep the individual and sensitive data of the involved parties private, while each party can get some knowledge from the output of the system without violating others’ privacy.

Neural network learning system is one of the important methods used in data mining and machine learning. However, to the best of our knowledge, there is no privacy-preserving technique to collaboratively produce a neural network in the case of two or more parties, in which each party has a private set of data. The only work in this area [18] deals with the case of client-server environment, and it is assumed that the neural network learning model already exists. There are different algorithms and architectures in neural networks such as perceptron learning algorithm, back-propagation model, and radial basis networks. In this paper we propose two new privacy-preserving protocols for perceptron learning algorithm, which is a feedforward neural network. These protocols are applied on distributed data environment, such that the original training data is vertically or horizontally partitioned among several parties and no party wants to reveal her/his own private data to the others.

In Section 2 neural networks are briefly reviewed along with background in privacy-preserving algorithm for neural networks. Two protocols for horizontally and vertically partitioned data are proposed in Section 3. Section 4 is dedicated to the conclusions and future work.

II. BACKGROUND AND RELATED WORK

Neural Networks system is an information processing concept which processes information in the same way as biological nervous systems. Some common applications of this system are classification, pattern recognition, function approximation and filtering. Normally, this system has a huge number of processing elements, which are highly interconnected and makes a model figuring out patterns or complex relationships between input and output data. The key point in this system is that it can learn and modify itself using its inputs. It is composed of a set of artificial neurons, or computational cells and a set of one-way connectors connecting those cells together [19]. Perceptron learning algorithm is a fundamental and important algorithm in feedforward neural networks.
learning systems proposed by Rosenblatt in [20]. Algorithm 1, adopted from [19], shows the steps of this method. The weights vector \( W = \langle W_0, W_1, \ldots, W_p \rangle \) for a neural network model with \( p \) inputs, is computed from the set of \( N \) training examples
\[
E = \{ \langle E^1, C^1 \rangle, \ldots, \langle E^N, C^N \rangle \}.
\]
Each \( E^k \) is a \( p \)-vector input such that
\[
E^k = \langle u_1, u_2, \ldots, u_p \rangle
\]
and it is extended to a \((p + 1)\)-vector by adding an additional input, \( u_0 \) with value +1 for every training example for the bias. \( C^k \) is the corresponding output of \( E^k \).

Perceptron learning algorithm is also used as a base algorithm for multi-layer constructive algorithms such as Tower and Pyramid algorithms. Therefore, after providing privacy-preserving protocols for perceptron learning algorithm, they can be extended to apply on other types of learning models with different configurations.

Barni et al. in [18] presented an asymmetric privacy-preserving protocol neural network for client-server environment. In this protocol, a client, or data owner, is able to process her/his data and receive the resulting output, using the neural network model owned by the server. In this approach, it was assumed that learning model has already been created by the server. They proposed three algorithms with different levels of privacy-preserving. In the first one, weights vector is private and the computation of weighted sum of the input data and weights vector is done securely. Parties use a secure dot product protocol to compute weighted sum, in which weights come from the server and inputs provided by the client. The client then applies the activation function, which is known to both parties, to the dot product result.

In the second algorithm, activation function is considered private for the server. Two types of functions were assumed. The first one was a threshold function which can be solved by using a secure comparison protocol. For the second type of functions, which can be approximated with a polynomial, Oblivious Polynomial Evaluation (OPE) was used to securely evaluate the private function. In the third algorithm, the server prevents the client from having a correct prediction of the underlying neural network model, by adding some fake cells to the system and resetting the outbound weights such that the final result remains unchanged.

One important privacy problem in these algorithms is that the client, after sending some request to the server, is able to learn the model. Also, in the second algorithm that the activation function is private, although OPE is used to hide the activation function, it can be disclosed to the client after receiving the result of some requests from the server.

Orlandi et al. [21] proposed another protocol which improves the number of interactions between the Neural network owner and data provider. It also prevents information leakage at the intermediate levels. All the intermediate computations in this protocol are concealed, and the terms of the comparison are obfuscated for the evaluation of activation function and are sent to the data owner by the neural network owner such that the real values are not revealed while the correct output can be calculated by the data owner. Homomorphic encryption, Paillier cryptosystem [22], and secure dot product are used in this protocol as building blocks.

Another work by Secretan et al. [23] presents a protocol for Probabilistic Neural Network (PNN) learning system using Bayesian theorem (Bayesian optimal classifier). They assume that the training data is known to all the parties, and is already loaded to the memory prior to the initiation of the PNNs performance phase, and each party can have private or public query for testing data to get the prediction of the class value. Also, because of using Secure Sum as a sub-protocol, it is assumed that at least three parties are involved in the protocol.

### III. New Protocols for Distributed Data

The main idea in the following protocols is to design privacy-preserving protocols for perceptron learning algorithm when input data is horizontally or vertically partitioned among multiple parties. In these protocols, in each step of the perceptron algorithm private output shares are created from the private input shares, and the final model is privately shared among the parties. Our protocols also cover both separable and non-separable datasets.

According to the definition provided by Goldreich [24], privacy means that each party can only get information which is inferred by using its own input and output available to that party. However, we believe that output share of a party should not help getting access to others’ private and sensitive information. Thus, in our protocol the final model is not released as a whole to each party. It is rather partitioned to private shares among the parties. Also, we assume that the parties are semi-honest. A semi-honest party properly follows the protocol, except that she/he might use received intermediate outputs to figure out some private information belonging to the other parties.

#### A. New Protocol For Horizontally Partitioned Data

In this section, we propose a protocol for creating a neural network learning model, using the perceptron algorithm, in which the data is horizontally partitioned among several parties. In horizontal or homogeneous distribution each party owns the value of all attributes of some records or rows of the whole database. At the end of this protocol, the model is privately shared among the parties involved and they can jointly and securely use the model to predict the output for a target data. The model considered here is a single-layer network using a threshold function, shown in Algorithm 1, as the activation function.

Suppose dataset \( D \) is horizontally partitioned to \( D_1, D_2, \ldots, D_m \) owned by parties \( P_1, P_2, \ldots, P_m \) respectively, and \( |D_i| = n_i, 1 \leq i \leq m \). Each item \( d_{i,j} \in D_i, 1 \leq j \leq n_i \), is a pair \( < E_{i,j}, C_{i,j} > \), in which \( E_{i,j} = \langle u_{i,j,1}, u_{i,j,2}, \ldots, u_{i,j,p} \rangle \) is the input vector, \( C_{i,j} \) is its corresponding output, and \( p \) is the number of input cells. Our goal is to compute the network weights
vector $W = \langle W_0, W_1, \cdots, W_p \rangle$ for this set of training example. To preserve the privacy of the learning model, this vector will be privately divided to all the parties such that $w_i = \langle w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle$ belongs to $P_i$ and:

$$W_k = \sum_{j=1}^{m} w_{j,k}, \quad 0 \leq k \leq p \quad (1)$$

At the beginning of the protocol, $W$ should be initialized to a vector with small values. Each party randomly and privately generates its own vector values. To make sure that the main vector $W$ has small values, parties agree to set their vector values in a way that the summation (1) for each $k$ should be a small number. For instance, parties with odd index start by a negative number and alternatively change the sign of the next value, and parties with even index do the opposite. Therefore, the summation of corresponding items would be a small value.

Now, each party $P_i$ has the following information:

$$d_{i,1} = \langle E_{i,1}, C_{i,1} \rangle$$
$$\vdots$$
$$d_{i,n_i} = \langle E_{i,n_i}, C_{i,n_i} \rangle$$

$E_{i,j} = \langle 1, w_{i,j,1}, w_{i,j,2}, \cdots, w_{i,j,p} \rangle$

$w_i = \langle w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle$.

Steps of the protocol are as follows:

1) **Selecting a party**: One party is randomly selected from the $m$ parties, say $P_i$.

2) **Selecting an item**: Party $P_i$ randomly generates an integer number $j$, $1 \leq j \leq n_i$, and selects item $d_{i,j}$.

3) **Computing weighted sum**: Now, $R = E_{i,j} \cdot W$ has to be computed.

$$R = E_{i,j} \cdot W = E_{i,j} \cdot \langle W_0, W_1, \cdots, W_p \rangle$$

$$= E_{i,j} \cdot \langle w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle + E_{i,j} \cdot \langle w_{2,0}, w_{2,1}, \cdots, w_{2,p} \rangle + \cdots$$

$$+ E_{i,j} \cdot \langle w_{m,0}, w_{m,1}, \cdots, w_{m,p} \rangle$$

In this equation, $E_{i,j} \cdot \langle w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle$ is computed locally by $P_i$ because both sides of the dot product belong to this party. The value of other dot products, i.e. $E_{i,j} \cdot \langle w_{k,0}, w_{k,1}, \cdots, w_{k,p} \rangle$, for $k \neq i$ is computed jointly by $P_i$ and $P_k$ using a secure dot product protocol, such as [25]. Thus, we have:

$$E_{i,j} \cdot \langle w_{k,0}, w_{k,1}, \cdots, w_{k,p} \rangle = R_{i,k} + R_k$$

in which $R_{i,k}$ and $R_k$ are private output shares of $P_i$ and $P_k$ respectively. Thus:

$$R = E_{i,j} \cdot W = (R_{i,1} + R_{i,2} + \cdots + R_{i,m}) + (R_1 + R_2 + \cdots + R_{l-1} + R_{l+1} + \cdots + R_m)$$

If we assume $R_i = R_{i,1} + R_{i,2} + \cdots + R_{i,m}$ then:

$$R = R_1 + R_2 + \cdots + R_m$$

(2)

4) **Applying activation function**: Threshold function is applied by comparing $\text{sign}(R)$ and $\text{sign}(C_{i,j})$. According to the algorithm 1, we don’t need to compute the result of the summation (2), and we can just run a sub-protocol to find $\text{sign}(R)$. If we convert this summation to multiplication of output shares, then by having the number of negative outputs, the sign of the summation can be determined. For this purpose, we first use the sub-protocol Secure Multi-party Addition, such that:

$$R = \sum_{l=1}^{m} R_l \prod_{l=1}^{m} r_l.$$ 

Then, each party $P_k$, $k \neq i$, sends the sign of its private output share, i.e. $\text{sign}(r_k)$, to $P_i$, and $P_i$ by counting the number of negative signs determines $\text{sign}(R)$ and compares it with $\text{sign}(C_{i,j})$, which already belongs to this party.

5) **Adjusting the weights vector**: If

$$\text{sign}(E_{i,j} \cdot W) = \text{sign}(C_{i,j})$$

then nothing is needed to do, otherwise $W$ has to be adjusted using $E_{i,j}$ and $C_{i,j}$ as follows:

$$W = W + C_{i,j} \cdot E_{i,j}$$

$$= \langle W_0, W_1, \cdots, W_p \rangle + C_{i,j} \cdot E_{i,j}$$

$$= \langle \underbrace{w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle +}$$

$$\vdots$$

$$\underbrace{w_{i-1,0}, w_{i-1,1}, \cdots, w_{i-1,p} \rangle +}$$

$$\underbrace{w_{i,0}, w_{i,1}, \cdots, w_{i,p} \rangle +}$$

$$C_{i,j} \cdot E_{i,j}$$

$$\vdots$$

$$\underbrace{w_{i+1,0}, w_{i+1,1}, \cdots, w_{i+1,p} \rangle +}$$

$$\vdots$$

$$\underbrace{w_{m,0}, w_{m,1}, \cdots, w_{m,p} \rangle}$$

As we see in this equation, only $P_i$ has to modify its weights vector by adding $C_{i,j} \cdot E_{i,j}$ to it, which is known to this party, and the weights vectors belonging to the other parties are not changed.

6) Go to step 1.

Although perceptron learning algorithm correctly works for separable set of training examples, it is not suitable for non-separable datasets. For nonseparable set of training examples other algorithms are used, such as Pocket algorithm [19], in which the weights vector with the longer run of correct classifications is kept inside the iteration. Thus, we modify our protocol to handle Pocket algorithm as well. Two integer variables, $\text{Run}_W$ and $\text{Run}_{W\text{Pocket}}$, have to be stored and updated inside the iteration. $\text{Run}_W$ is the number of items correctly classified by the current weights vector and $\text{Run}_{W\text{Pocket}}$ is
the number of items correctly classified by the pocket weights vector, which can be public and known to all the parties.

Also, a weights vector, $W_{\text{Pocket}}$ is considered to keep the weights vector with the longest run of the correct classifications, and is privately shared to the parties in the same way of the current weights vector, $W$. Before starting the protocol $\text{Run}_W$ and $\text{Run}_{W_{\text{Pocket}}}$ are set to 0, and $W_{\text{Pocket}}$ is set to $W$, it means that each party has its own private share of $W_{\text{Pocket}}$. Inside the protocol and in step 4, if $\text{sign}(E_{ij} \cdot W) = \text{sign}(C_{ij})$, $\text{Run}_W$ is increased by one, and if $\text{Run}_W > \text{Run}_{W_{\text{Pocket}}}$, each party replaces its own share of $W_{\text{Pocket}}$ with its current share of $W$, and $\text{Run}_{W_{\text{Pocket}}}$ is set to $\text{Run}_W$. By this modification, and by having a specific threshold for the desired number of items that are correctly classified by a weights vector, algorithm is able to find and keep the best weights vector in case of nonseparable set of training data.

For security analysis, we use the composition theorem of secure protocols by going through the steps of the protocol. This theorem states that if all sub-protocols used in a protocol are privacy-preserving, such that intermediate output of one sub-protocol becomes input of the next sub-protocol as a private share, then the main protocol is also privacy-preserving [26], [27]. First of all, each party securely and randomly generates its own initial weights vector and thus it is not known by the other parties. In step 2 the selected party randomly selects an item from its dataset. Therefore other parties do not know the input vector values. In step 3, secure dot product is applied to each pair of private vectors belonging to two parties and final result is divided to two private shares for the parties involved. Goethals et al., in [25], have proposed an efficient protocol for this secure two-party computation by using homomorphic encryption and we use that in our protocol. Thus, both sides are unaware from each other’s input and output. In step 4, because the summation is converted to multiplication of private shares and parties only send the sign of their outputs to the party $P_i$, their private values, i.e., $R_i$’s, are not revealed. We use our algorithms proposed in [17] for secure multi-party addition in this protocol. By using this building block, each party $P_i$ having private inputs $x_i$, obtains a private output share $r_i$ such that:

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} r_i$$

Finally in step 5, only $P_i$’s private share of the weights vector is locally modified by this party and no information is exchanged. Therefore, the final shares of the weights vector are kept private.

**B. New Protocol For Vertically Partitioned Data**

In this section, a protocol is presented to produce neural network model using perceptron learning algorithm when data is vertically partitioned among the parties. It means each party owns a subset of attributes, and class attribute or the output $C_i$ is public. Same as the protocol for horizontal case, learning model is finally shared among the parties involved, and prediction of testing data is jointly and securely done by all the parties.

Suppose dataset $D$ is vertically partitioned to $D_1, D_2, \ldots, D_m$ owned by parties $P_1, P_2, \ldots, P_m$ respectively, and the number of attributes in $D_i$ is $n_i$, such that $\sum_{i=1}^{m} n_i = p$, in which $p$ is the number of all attributes. Each item $d_i \in D$ is a pair $(E_i, C_i)$, in which:

$$E_i = < 1, u_{1,1}, u_{1,2}, \ldots, u_{1,n_1}, u_{2,1}, u_{2,2}, \ldots, u_{2,n_2}, \ldots, u_{m,1}, u_{m,2}, \ldots, u_{m,n_m} >$$

is the input vector and $C_i$ is its corresponding output. Note that $u_{1,1}, u_{1,2}, \ldots, u_{1,n_1}$, and $u_{i,n_i}$ belong to $P_i$ and let the first item of $E_i$, 1, is owned by the first party, $P_1$. Our goal is to compute the network weights vector $W = < W_0, W_1, \ldots, W_p >$ for this set of training examples. To preserve the privacy of the computed learning model, this vector will be securely distributed to all the parties, such that:

$$W = < w_{1,0}, w_{1,1}, w_{1,2}, \ldots, w_{1,n_1}, w_{2,1}, w_{2,2}, \ldots, w_{2,n_2}, \ldots, w_{m,1}, w_{m,2}, \ldots, w_{m,n_m} >$$

Therefore, each party knows the initial, intermediate, and final weight values for its own attributes. Without loss of generality, we assume that $W_0 = w_{1,0}$ is maintained by $P_1$. First, $W$ should be initialized to a vector with small values. Thus, each party randomly and privately generates its own part of the weights vector, i.e. $P_i$ initializes $w_{i,1}, w_{i,2}, \ldots$, and $w_{i,n_i}$.

Following are the steps of the protocol:

1) **Selecting an item**: One item $d_i$ is randomly selected from the set of training examples

2) **Computing weighted sum**: $R = E_i \cdot W$ is computed:

$$R = E_i \cdot W = < 1, u_{1,1}, \ldots, u_{1,n_1}, u_{2,1}, \ldots, u_{2,n_2}, \ldots, u_{m,1}, \ldots, u_{m,n_m} >$$

$$= < w_{1,0}, w_{1,1}, \ldots, w_{1,n_1}, w_{2,1}, \ldots, w_{2,n_2}, \ldots, w_{m,1}, \ldots, w_{m,n_m} >$$

$$= < 1, u_{1,1}, \ldots, u_{1,n_1} > < w_{1,0}, \ldots, w_{1,n_1} > + < u_{2,1}, \ldots, u_{2,n_2} > < w_{2,1}, \ldots, w_{2,n_2} > + \ldots$$

$$= R_1 + \ldots + R_m$$

In this equation, $R_i$ is locally computed by the party $P_i$.

3) **Applying activation function**: $\text{sign}(R)$ and $\text{sign}(C_{ij})$ have to be compared. Same as the previous protocol, we don’t need to compute $R$, and only $\text{sign}(R_i)$ has to be compared with $\text{sign}(C_{ij})$, by using Secure Multi-party Addition same as that in the previous protocol. Then, one party, say $P_k$, is selected and each party $P_j$, $j \neq k$, sends the sign of its output, to $P_k$. Now, $P_k$ by counting the number of negative signs determines the sign of the summation and compares it with $\text{sign}(C_{ij})$. 
4) Adjusting the weights vector: If

$$\text{sign}(E_i \cdot W) = \text{sign}(C_i)$$

we do nothing, otherwise $W$ has to be adjusted using $E_i$ and $C_i$ as follows:

$$W = W + C_i \ast E_i$$

$$= < w_{i,0}, w_{i,1}, \cdots, w_{i,n_1}, \cdots, w_{i,m,1}, \cdots, w_{i,m,n_m} > +$$

$$C_i < 1, u_{i,1}, \cdots, u_{i,n_1}, \cdots, u_{i,m,1}, \cdots, u_{i,m,n_m} >$$

$$= < w_{i,0} + C_i, w_{i,1} + C_i \ast u_{i,1}, \cdots, w_{i,n_1} + C_i \ast u_{i,n_1}, \cdots, w_{i,m,1} + C_i \ast u_{i,m,1}, \cdots, w_{i,m,n_m} + C_i \ast u_{i,m,n_m} >$$

Thus, each party modifies its own part of the weights vector.

5) Go to step 1.

At the end, each party has the corresponding weights to its input attributes from the weights vector. Modification of this protocol to handle the Pocket algorithm is straightforward.

We go through the protocol to check that the parties’ privacy is preserved during the algorithm. In the initialization step, each party, according to its attributes, randomly and privately generates its own part of the weights vector. In step 2, each dot product is computed locally by each party and no information is exchanged. In step 3, after running secure multi-party addition protocol, each party $P_i$, $i \neq k$, only sends the sign of its private output to $P_k$ and no information about its input, i.e. $R_i$, is revealed. Finally in step 4, each party locally modifies its own part of the weights vector.

Both of this and previous protocols for vertical and horizontal cases can be applied incrementally. It means, whenever the model needs to be adjusted with extra training data, we can use the constructed model which is shared among the parties and apply the same protocol to adjust the weights vector according to the new data.

IV. CONCLUSIONS AND FUTURE WORK

Nowadays, Neural network is one of the important subjects in machine learning and data mining systems, and in most of the underlying applications, such as health, government, and commercial, privacy preserving is very crucial. Therefore, existing algorithms in this learning system need to be investigated and privacy-preserving protocols have to be proposed. In this paper two new protocols are presented for perceptron learning algorithm in multi-party environment when data is horizontally or vertically partitioned. To preserve the privacy of the output model as well as the input data, final weights vector is not released to all the parties involved. Instead, each party has a private share of the model, and they can jointly use the model to predict the output of the target data.

Neural network has many different algorithms to create various models according to the type of data, and activation functions. Thus, a possible future work is to extend the proposed protocols in this paper to cover other types of configurations, such as continuous data, multi-layer models, stochastic activation functions like sigmoid function, back-propagation algorithm, and recurrent networks. Currently, we are working on the back-propagation algorithm with sigmoid function as the activation function on continuous input and output data, to design privacy-preserving protocols for different types of data distributions in multi-party environment.

REFERENCES


