Batch: SC_1 **Roll No.: 1911027**

Experiment No. 09

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Case study on any one Special Neural Networks

Aim : Explain Privacy-Preserving Backpropagation Neural Network.

Expected Outcome of Experiment:

CO3 : Analyze various neural network architectures

Books/ Journals/ Websites referred:

- J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
- Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
- S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI, 2003.
- http://library.thinkquest.org/C007395/tqweb/history.html

Pre Lab/ Prior Concepts:

1. **Basics of neural networks.**
2. **Backpropagation networks.**

Detailed study on Special Neural Network:

Privacy-Preserving Backpropagation Neural Network

A distributed system, also known as distributed computing, is a system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user. Distributed computing is a model in which components of a software system are shared among multiple computers. Even though the components are spread out across multiple computers, they are run as one system. This is done in order to improve efficiency and performance. There are various advantages of distributed computing:

- Fault Tolerance and Redundancy
- Low Latency
- Cost Effectiveness
- Efficiency

Hence, keeping in mind the above advantages, nowadays, many machine learning problems now have to deal with distributed input data. One of the major problems that arrive with this new approach is that of privacy. Hence, it is important to address the privacy concern of each data holder by extending the privacy preservation notion to original learning algorithms**.** In this paper, they have focused on preserving the privacy of data and the parties (provider of the data) in a multilayer neural network. They have presented a two-party distributed algorithm of backpropagation which allows a neural network to be trained without requiring either party to reveal her data to the other. For example, if medical researchers want to apply machine learning to study health care problems, they need to collect the raw data from hospitals and the follow-up in-formation from patients. Then, the privacy of the patients must be protected, according to the privacy rules in Health Insurance Portability and Accountability Act (HIPAA), which establishes the regulations for the use and disclosure of Protected Health Information. Now, a fundamental question arises; why would the researchers want to build a learning model without first collecting all the training data on one computer? In many real-world cases, it is rather difficult to find such a trusted learner, since some data holders will always have concerns. Moreover, distributed and networked computing environments now-a-days, collaborations will greatly benefit scientific advances. In this paper, they have focused on multilayer neural networks, in which the privacy preservation problem is far from being practically solved. A preliminary approach is proposed to enable privacy preservation for gradient-descent methods in general. However, in terms of multilayer neural networks, their protocol is limited as it is only for one simple neural network configuration with one node in the output layer and no hidden layers. Although their protocol is elegant in its generality, it may be very restricted in practice for privacy-preserving multilayer neural networks.

Multilayer networks have a hidden layer, whose neurons are not directly connected to the output. They are used for linearly inseparable problems. In this paper, they have considered a neural network of three layers, where the hidden-layer activation function is sigmoid and the output layer is linear. The multilayer neural network's configuration is given as a – b – c. So there's one hidden layer whose nodes are represented by {h1, h2,.. }. Input vector is given by {x1, x2..} and output vector is given as {o1, o2, …} . They have used a backpropagation method as it's a multilayer neural network. Also, they have used MSE (Mean Square error) as the error function. Backpropagation involves taking partial derivatives of and the equations are as follows:

$$\frac{\partial e}{\partial w_{ij}^{o}} = -(t_i - o_i)h_j \qquad (1)$$

$$\frac{\partial e}{\partial w_{jk}^{h}} = -h_j(1-h_j)x_k \sum_{i=1}^{c}[(t_i - o_i)w_{ij}^{o}]. \qquad (2)$$

Then the concept of piecewise linear approximation of activation function is introduced. The major reason for introducing the approximation is that 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. Also, they have adopted the semi-honest model. When computing function 'f' is in a distributed fashion, a semi honest 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 they receive during the execution. In order to guarantee the security of distributed algorithms of computing, it must be ensured that each party can learn nothing beyond what can be implied by her own input and output. Semi Honest model is a right fit for, because normally participants want to learn the neural network learning results and thus they are willing to follow the algorithm to guarantee the results correctness. The security guaranteed in a semi honest model can relieve the concerns about their data privacy. They have presented a privacy-preserving distributed algorithm for training the neural networks with backpropagation algorithm. A privacy-preserving testing algorithm can be easily derived from the feedforward part of the privacy-preserving training algorithm.

Privacy-Preserving Neural Network Training Algorithm:

I will be discussing the Privacy-preserving distributed algorithm for the neural network training process under the assumption that we already have the algorithm to securely compute the piecewise linear function. privacy-preserving distributed algorithm for the neural network training process under the assumption that we already have the algorithm to securely compute the piecewise linear function. 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 parties can imply the original data information from the intermediate results. Here by "randomly share," I mean that each party holds a random number and the sum of the two random numbers equals the intermediate result.

Algorithm: Privacy-Preserving Distributed Algorithm for Backpropagation Training:

Initialize all weights to small random numbers, and make them known to both parties.

Repeat

for all training sample $\langle\{x_A, x_B\}, t(x)\rangle$ **do**

Step 1: feedforward stage

1.1) **For each** hidden layer node h_j, party A computes $\sum_{k \le m_A} w_{jk}^h x_k$, and party B computes $\sum_{m_A < k \le m_A + m_B} w_{jk}^h x_k$.

1.2) Using Algorithm 2, 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}, respectively, s.t. $h_{j1} + h_{j2} = f(\sum_k w_{jk}^h x_k)$.

1.3) **For each** output layer node o_i, party A computes $o_{i1} = \sum_i w_{ij}^o h_{j1}$ and party B computes $o_{i2} = \sum_i w_{ij}^o h_{j2}$, s.t. $o_i = o_{i1} + o_{i2} = \sum_i w_{ij}^o h_{j1} + \sum_i w_{ij}^o h_{j2}$.

Step 2: backpropagation stage

2.1) **For each** output layer weight w_{ij}^o, parties A and B apply Algorithm 3 to securely compute the product $h_{j1}o_{i2}$, obtaining random shares r_{11} and r_{12}, respectively, s.t. $r_{11} + r_{12} = h_{j1}o_{i2}$. Similarly, they compute the random partitions of $h_{j2}o_{i1}$, r_{21}, and r_{22}, s.t. $r_{21} + r22 = h_{j2}o_{i1}$. Party A computes $\Delta_1 w_{ij}^o = (o_{i1} - t_i)h_{j1} + r_{11} + r_{21}$ and party B computes $\Delta_2 w_{ij}^o = (o_{i2} - t_i)h_{j2} + r_{12} + r_{22}$.

2.2) **For each** hidden-layer weight w_{jk}^h, using Algorithm 1.1, parties A and B jointly compute $\sum_i[-(t_i - o_i)w_{ij}^o]h_j(1 - h_j)$, obtaining random shares q_1 and q_2, respectively, s.t. $q_1 + q_2 = \sum_i[-(t_i - o_i)w_{ij}^o]h_j(1 - h_j)$. If $k \le m_A$, that is, A holds the input attribute x_k, applying Algorithm 3 to securely compute $x_k q_2$, A and B, respectively, get r_{61} and r_{62}, s.t. $x_k q_2 = r_{61} + r_{62}$.

Then, $\Delta_1 w_{jk}^h = q_1 x_k + r_{61}$ and $\Delta_2 w_{jk}^h = r_{62}$. If $m_A < x_k \leq m_A + m_B$, A and B apply Algorithm 3 to get r_{61} and r_{62}, s.t $x_k q_1 = r_{61} + r_{62}$. In this case, $\Delta_1 w_{jk}^h = r_{61}$, $\Delta_2 w_{jk}^h = q_2 x_k + r_{62}$. A and B, respectively, compute $\Delta_1 w_{jk}^h$, $\Delta_2 w_{jk}^h$.

Step 3: A (B, resp.) sends $\Delta_1 w_{ij}^o$ and $\Delta_1 w_{jk}^h$ ($\Delta_2 w_{ij}^o$ and $\Delta_2 w_{jk}^h$, resp.) to B (A, resp.). A and B compute $w_{ij}^o \leftarrow w_{ij}^o - \eta(\Delta_1 w_{ij}^o + \Delta_2 w_{ij}^o)$ for each hidden-layer weight, and $w_{jk}^h \leftarrow w_{jk}^h - \eta(\Delta_1 w_{jk}^h + \Delta_2 w_{jk}^h)$ for each output-layer weight.

end for

Until (termination condition)

Algorithm for computing piecewise linear approximated sigmoid function:

Step 1) Party A generates a random number R and computes $y(x_1 + i) - R$ for each i, s.t. $-n < i \leq n$. Define $m_i = y(x_1 + i) - R$. Party A encrypts each m_i using ElGamal scheme and gets $E(m_i, r_i)$, where each r_i is a new random number. Party A sends each $E(m_i, r_i)$ in the increasing order of i.

Step 2) Party B picks $E(m_{x_2}, r_{x_2})$. She rerandomizes it and sends $E(m_{x_2}, r')$ back to A, where $r' = r_{x_2} + s$, and s is only known to party B.

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 (by doing partial decryption on the already partially decrypted message) to get $m_{x_2} = y(x_1 + x_2) - R$. Note R is only known to A and m_{x_2} is only known to B. Furthermore, $m_{x_2} + R = y(x_1 + x_2) = f(x)$.

Moving on to how these algorithms protect privacy of both the parties, there are two places in the algorithms where approximation for the goal of privacy has been introduced. One is that the sigmoid function used in the neuron computing is replaced by a piecewise linear function. The other approximation is introduced by mapping the real numbers to fixed-point representations to enable the cryptographic operations in Algorithms 2. This is necessary in that intermediate results, for example, the values neurons are represented as real numbers in normal neural network learning, but cryptographic operations are on discrete finite fields.

Privacy analysis: Recall that in a semi honest model, the parties follow the protocol and may try to analyze what she can see during the protocol execution. To guarantee security in the semi honest model, we must show that parties can learn nothing beyond their outputs from the information they get throughout the protocol process.4 A standard way to show this is to construct a simulator which can simulate what the party can see in the protocol given only the input and the output of the protocol for this party.

Although, there is a chance of accuracy loss caused by the fixed-point representation.

- Error in Truncation
- Error in feedforward stage
- Error in Output-Layer Delta
- Error in Hidden -Layer Delta
- Error in Weight Update

Moving on to measuring the accuracy and overhead of the algorithms presented in this paper, below are the datasets and the parameters for the backpropagation multilayer neural network:

TABLE I
DATA SETS AND PARAMETERS

Dataset	Sample	Class	Architecture	Epochs	Learning Rate
kr-vs-kp	3196	2	$36-15-1$	20	0.1
Iris	150	3	$4-5-3$	80	0.1
diabetes	768	2	$8-12-1$	40	0.2
Sonar	104	2	$60-6-2$	150	0.1
Landsat	6435	6	$36-3-6$	12	0.1

They have measured the accuracy loss of our privacy-preserving algorithms when the neural network is trained with a fixed number of epochs (shown in Table I). The number of epochs set is based both on the number of examples and on the parameters (i.e., topology) of the network.

Specifically, they have used 80 epochs for small problems involving fewer than 250 examples; 40 epochs for the midsized problems containing between 250 to 500 examples; and 20 epochs for larger problems. Table II shows testing set error rates for both non privacy-preserving backpropagation neural network and privacy-preserving backpropagation neural networks.

TABLE II
TEST ERROR RATES COMPARISON

Dataset	Non-privacy-preserving Version	Privacy-preserving Algorithm 1
kr-vs-kp	12.5%	15.5%
Iris	14.17%	19.34%
Pima-indian-diabetes	34.71%	38.43%
Sonar	18.26%	21.42%
Landsat	4.12%	5.48%

In this paper, researchers present a privacy-preserving algorithm for backpropagation neural network learning. The algorithm guarantees privacy in a standard cryptographic model, the semi honest model. Although approximations are introduced in the algorithm, the experiments on real-world data show that the amount of accuracy loss is reasonable. Using the techniques followed in the paper, it should not be difficult to develop the privacy-preserving algorithms for BPN learning with three or more participants. In this paper, we have considered only the backpropagation neural network.

<u>Conclusion:</u> Hence, this paper presented a privacy-preserving algorithm for backpropagation neural network learning. The algorithm guarantees privacy in a standard cryptographic model, the semi honest model. Although approximations are introduced in the algorithm, the experiments on real-world data show that the amount of accuracy loss is reasonable. Thus we successfully study special neural networks.

Date: 24 / 11 /2021 **Signature of faculty in-charge**