Simplified security learning using vertically partitioned data with IoT

Hirofumi Miyajima\textsuperscript{1a)}, Noritaka Shigei\textsuperscript{2b)}, Hiromi Miyajima\textsuperscript{2c)}, and Norio Shiratori\textsuperscript{3d)}

\textsuperscript{1} School of Information and Data Sciences, Nagasaki University, 1–14 Bunkymachi, Nagasaki city, Nagasaki 852–8521, Japan
\textsuperscript{2} Kagoshima University, 21–40 Korimoto 1, Kagoshima city, Kagoshima 890–0065, Japan
\textsuperscript{3} Chuo University, 13–27 Kasuga 1, Bunkyoku, Tokyo 112–8551, Japan

\textsuperscript{a)} miyajima@nagasaki-u.ac.jp
\textsuperscript{b)} shigei@eee.kagoshima-u.ac.jp
\textsuperscript{c)} k2356323@kadai.jp
\textsuperscript{d)} norio@riec.tohoku.ac.jp

Received January 3, 2021; Revised March 11, 2021; Published July 1, 2021

Abstract: Edge (or fog) computing is known as a method for improving the conventional cloud system. The basic idea is to consider a system that places edges (servers) between the cloud and the terminals (things). Then, how should machine learning be realized on the edge system? Fast and secure learning methods are desired for machine learning. Secure systems using distributed processing have attracted attention. SMC (Secure Multiparty Computation) is one of the typical models to realize secure learning. Horizontally and vertically partitioned data are known for SMC. The latter is a method consisting of dividing the dataset into element-separated subsets. It is desired to develop a method for directly executing learning using element-separated subsets. Vertically partitioned data (VPD) is considered to be a data structure that realizes such learning. In the previous papers, we proposed learning methods for BP (Back Propagation) and NG (Neural Gas) using VPD. There, we did not consider about the amount of data transferred between servers. In this paper, simplified learning methods that eliminate wasteful data transfer compared to the method in the previous papers are proposed, and its effectiveness is shown. That is, the data transfer from the central server to each server was reduced to $1/L$, where $L$ is the number of training data.

Key Words: IoT, machine learning, secure multiparty computation, batch processing, back propagation, edge computing, vertically partitioned data
1. Introduction

Cloud computing, one of the basic technologies that support ICT or AI, is used in various fields. However, with the transition to the Internet of Things (IoT), the number of servers (things) connected to the cloud system increases. Therefore, the load on the servers increases and the processing power of the cloud system is significantly reduced. It leads to significant processing time delay. Edge (or Fog) computing is known as a method for improving the conventional cloud system [1, 2]. This is a model in which the main servers located far in the conventional system are replaced with local servers located near terminals and things. It means a short cut for the tasks for the main servers. Remark that edge system servers are less powerful than cloud servers. Therefore, the problem with the calculation by the edge system is how to realize complicated calculation while maintaining the confidentiality of data by combining multiple servers with low capacity. Then, how should machine learning be realized on the edge system while protecting personal information? SMC is one of the typical models to realize machine learning safety [3–5]. The basic idea of SMC is to implement the data processing by dividing a dataset into subsets, processing the subset on each server, and integrating the results. This is a way to keep the dataset safe. In particular, for machine learning using SDM (Steepest Descent Method), secure learning methods can be realized. Horizontally partitioned data (HPD) and VPD are known for SMC [4, 5]. The former is a method consisting of dividing a dataset into subsets, and the latter is a method consisting of dividing the dataset into element-separated subsets. Both methods are ones of updating the parameters while learning data does not go out. Federated Learning (FL) is considered to be one of the learning methods using HPD for edge computing [6]. Some methods have been proposed for realizing machine learning on cloud and edge systems using SMC so far. In particular, for HPD, several methods have been proposed, including FL [7]. On the other hand, for VPD, few learning methods have been proposed. In the previous papers, the authors proposed learning methods for Back Propagation (BP) and Neural Gas (NG) using VPD. In general, learning method using SMC consists of calculation and data transfer. Therefore, if the data transfer can be reduced, the load on each server can be reduced. In this paper, we propose simplified learning methods that reduce data transfer of the BP and NG learning methods proposed in the previous papers. In Section 2, secure computation, the model, data structure, and learning method of BP and NG used in this paper will be explained as preparation. In Section 3.1, BP proposed in the previous paper and its simplified learning methods are presented. In Section 3.2, NG proposed in the previous paper and its simplified NG learning methods are presented. In Section 3.3, the relation between the proposed and the conventional methods is shown. In Section 4, the accuracy of the conventional method of online and batch calculation using one server and the accuracy of the two methods of Section 3 are compared by numerical simulation. Section 5 is a summary.

2. Preliminaries

2.1 Secure computation and a configuration for the edge system

Regarding data processing on cloud systems, it is desired to develop a technology for performing calculation processing while maintaining the confidentiality of data. From the viewpoint of privacy preserving of learning data for machine learning, the following studies are being conducted:

1) Secret sharing + SMC,
2) Homomorphic encryption [11],
3) Federated learning.

In these cases, 1) and 2) are methods based on encryption, and 3) is a method in which data is distributed and learning is performed by distributed computing. Each has its strengths and weaknesses. Method 3) divides the dataset into multiple subsets by using HPD and performs machine learning using distributed system so that the dataset is not concentrated in one sever. Therefore, it is desired that its security is preserved. Method 3) is also considered to be suitable for application to edge systems. The proposed method in this paper uses VPD instead of HPD. In the following, we will consider how to realize machine learning by distributed processing while preserving security using a separation method of data called VPD.

First, let us introduce an edge system that realizes distributed processing. One of the purposes of
Edge computing is combining multiple servers (edges) with low capabilities to build a high-performance system. In the following, an effective method to realize machine learning with such property on the edge system is proposed.

Figure 1(a) shows the conventional cloud system. Figure 1(b) shows a system of edge computing. The system is composed of main servers of the cloud and multiple servers (called edges) connected to the close distance from the terminals (things). Each edge is connected directly to each terminal. The data provided to the terminal is sent to each edge and processed. Even if the learning is performed as it is in Fig. 1(b), the security of the data is not always preserved. The problem is how to share and distribute data among the edges in order to execute effective computation while maintaining safety.

Figure 2 shows an edge system using vertically partitioned data, where the edge system is composed of $N+1$ edges and $N$ Terminals. $N+1$ edges are called Edge 0, Edge 1, ···, Edge $N$. The system is a network of one central server Edge 0 (also called an aggregator in other paper) and $N$ servers. In the model using VPD, learning data are decomposed into element-separated subsets. In this paper, for simplicity, it is assumed that one element of data is assigned to one server. Therefore, the number of servers and the number of dimensions of the data are equally $N$. If multiple elements are assigned to one server, the number of servers is smaller than the number of dimension of the data.

Let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$ for a positive integer $i$. An edge system works after each edge receives data from terminals as follows:

1) Each edge processes learning data and updates model parameters. To minimize the target function, each edge uses SDM to adjust the model parameters. Further, it sends model parameters to Edge 0.

2) Edge 0 collects partial computation for each edge, integrates using these results and send them
back to each edge for the next updating of parameters.

In this case, it is important that learning data does not go out from each edge, and only the training parameters move back and forth between Edge 0 and each edge.

2.2 BP neural network learning [7, 9]

Let \( x^l \in \mathbb{R}^N \) for \( l \in \mathbb{Z}^L \) and \( d : \mathbb{R}^N \rightarrow J \), where \( \mathbb{R} = (-\infty, \infty) \) and \( J = [0, 1] \). For the sets \( D = \{ (x^l, d(x^l)) \mid x^l \in J^N, l \in \mathbb{Z}^L \} \), \( D_{in} = \{ x^l \mid l \in \mathbb{Z}^L \} \) and \( D_{out} = \{ d(x^l) \mid l \in \mathbb{Z}^L \} \) of learning data, let us determine parameters of the three-layered neural network identifying learning data by BP method, where \( d(x^l) \) means the desirable output for input data \( x^l \) and \( L \) is the number of learning data. Let \( h = h_2 \circ h_1 : \mathbb{R}^N \rightarrow J \) for \( h_1 : \mathbb{R}^N \rightarrow J^M \) and \( h_2 : J^M \rightarrow J \). Let \( N \) and \( M \) be the numbers of elements for the first and second layers, respectively. Let \( w_i = (w_{i0}, w_{i1}, \ldots, w_{iN}) \) for \( i \in \mathbb{Z}^M \) and \( v = (v_0, v_1, \ldots, v_M) \) be weights for the second and output layers, respectively. Then \( h_1 \) and \( h_2 \) are as follows (See Fig. 3):

\[
y_i = h_1_i(x) = \tau \left( \sum_{j=0}^{N} w_{ij} x_j \right), \tag{1}
\]

\[
x_0 = 1, \quad h_1 = (h_{11}, \ldots, h_{1i}, \ldots, h_{1M})
\]

\[
\tau(u) = \frac{1}{1 + \exp(-u)}
\]

where

\[
x = (x_0, x_1, \ldots, x_j, \ldots, x_N) \in \mathbb{R}^{N+1}
\]

\[
y = (y_0, y_1, \ldots, y_i, \ldots, y_M) \in J^{M+1}
\]

and \( w_{i0} \) means the threshold value for the \( i \)-th neuron of the second layer.

Further,

\[
h_2(y) = \tau \left( \sum_{i=0}^{M} v_i y_i \right), \tag{2}
\]

\[
y_0 = 1,
\]

where \( v_0 \) means the threshold value of the output layer.

![Fig. 3. The relation between the dataset and three-layered neural network: Each element \( x^j \) of the set \( D_j \) for \( j \in \mathbb{Z}^N \) is provided to the \( j \)-th element of the first layer.](image)

Then, the evaluation function is defined as follow:

\[
E = \frac{1}{2L} \sum_{i=1}^{L} \left( h(x^i) - d(x^i) \right)^2 \tag{3}
\]
The weights \( w \) and \( v \) are updated based on the BP method as follows [9]:

\[
\triangle w_i(x') = -\alpha_1 \delta_2(x') h_i(x') \quad \text{for } i \in Z_M^* \tag{4}
\]

\[
\triangle v_i(x') = -\alpha_2 \delta_1(x') x_i \quad \text{for } j \in Z_N^* \text{ and } i \in Z_M \tag{5}
\]

where \( \alpha_1 \) and \( \alpha_2 \) are learning coefficients,

\[
\delta_2(x) = (h(x) - d(x)) h(x) (1 - h(x)) \tag{6}
\]

and

\[
\delta_1(x) = \delta_2(x) v_i h_i(x) (1 - h_i(x)) \quad \text{for } i \in Z_M. \tag{7}
\]

In batch learning, the following equations are used instead of Eqs. (4) and (5).

\[
\Delta v_i = -\sum_{l=1}^{L} \triangle v_i(x') \quad \text{for } i \in Z_M^* \tag{8}
\]

\[
\Delta w_{ij} = -\sum_{l=1}^{L} \triangle w_{ij}(x') \quad \text{for } j \in Z_N^* \text{ and } i \in Z_M \tag{9}
\]

### 2.3 Neural Gas method [8, 10]

Vector quantization techniques encode a data space, e.g., a subspace \( X \subseteq \mathbb{R}^d \), utilizing only a finite set \( W = \{w_i \mid i \in Z_r\} \) of reference vectors (also called cluster centers), where \( d \) and \( r \) are positive integers.

Let the winner vector \( w_i(x) \) be defined for any vector \( x \in X \) as follows:

\[
i(x) = \arg \min_{i \in Z_r} ||x - w_i|| \tag{10}
\]

where \( i(x) \) is an element of \( Z_r \) and represents the subscript of the element of \( W \) closest to the data \( x \), where \( |W| = r \). In Eq. (10), if there are some subscripts on the right side, select the smallest subscript.

From the finite set \( W \), \( X \) is partitioned as follows:

\[
X_i = \{x \in X \mid ||x - w_i|| \leq ||x - w_l|| \text{ for } l \in Z_r\} \tag{11}
\]

The sets \( X \) and \( W \) are called sets of input and reference vectors, respectively. There are convergence-guaranteed algorithms as Vector Quantization such as \( k \)-means, NG, Maximum Entropy and Fuzzy \( c \)-means. In the following, NG is introduced based on Ref. [10].

For NG method, the following method is used:

Given an input vector \( x \), we determine the neighborhood-ranking \( w_{ik} \) for \( k \in Z_{r-1}^* \), being the reference vector for which there are \( k \) vectors \( w_j \) with

\[
||x - w_j|| < ||x - w_{ik}|| \tag{12}
\]

Let \( \alpha \in [0, 1] \) and \( \lambda > 0 \).

If the number \( k \) associated with each vector \( w_i \) is denoted by \( k_i(x, w_i) \), then the adaptation step of online for adjusting the \( w_i \)'s is given by

\[
\Delta w_i = \alpha h_\lambda(k_i(x, w_i))(x - w_i) \tag{13}
\]

\[
h_\lambda(k_i(x, w_i)) = \exp \left( -k_i(x, w_i)/\lambda \right) \tag{14}
\]

\[
\alpha = \alpha_{int} \left( \frac{\alpha_{fin}}{\alpha_{int}} \right)^{\frac{T_{max}}{T_{max}}}
\]

where \( T_{max}, \alpha_{int}, \alpha_{fin} \) and \( \lambda \) mean the maximum number of epoch, initial and final learning coefficients, and the decay constant. The following function is used as an evaluation one:
\[
E = \sum_{\omega_i \in W, x \in X} \frac{h_\lambda(k_i(x, \omega_i))}{\sum_{\omega_j \in W} h_\lambda(k_j(x, \omega_j))} ||x - \omega_i(x)||^2
\]  

(15)

The number \( \lambda \) is called decay constant.

**Learning Algorithm: Neural Gas**

*Input*: The set \( X \) of input vectors.

*Output*: The set \( W \) of reference vectors.

**Step 1**: The initial values of reference vectors are set randomly. The learning coefficients \( \alpha_{int} \) and \( \alpha_{fin} \) are set. Let \( T_{\text{max}} \) be the maximum number of epoch.

**Step 2**: Let \( t = 1 \).

**Step 3**: Give a data \( x \in X \) randomly and each neighborhood-ranking \( k_i(x, \omega_i) \) is determined for \( i \in \mathbb{Z}_r \).

**Step 4**: Each reference vector \( \omega_i \) for \( i \in \mathbb{Z}_r \) is updated based on Eq. (13)

**Step 5**: If \( t \geq T_{\text{max}} \), then the algorithm terminates and the set \( W = \{ \omega_i \mid i \in \mathbb{Z}_r \} \) of reference vectors for \( X \) is obtained else go to Step 3 and \( t \leftarrow t + 1 \).

Similarly, the adaptation step for batch is given by replacing Eq. (13) with the following equation (16) as follows:

\[
\Delta \omega_i = \sum_{x \in X} \alpha h_\lambda(k_i(x, \omega_i))(x - \omega_i)
\]  

(16)

### 3. Learning algorithms for secure multiparty computation with IoT

This section proposes simplified algorithms for BP and NG using VPD proposed in Refs. [7] and [8]. The key idea is to perform calculation on Edge 0 as much as possible. This makes it possible to significantly reduce data transfer. Section 3.1 proposes a simplified algorithm of the BP method proposed in Ref. [7]. Section 3.2 proposes a simplified algorithm of the NG method proposed in Ref. [8]. Section 3.3 shows the relation between the conventional and proposed methods.

#### 3.1 Simplified BP learning for vertically partitioned data with IoT

Firstly, secure BP neural network learning proposed in Ref. [7] on vertically partitioned data with edge system is presented (See Table I). Each element-separated subset is distributed to each Edge (See Fig. 2). The \( j \)-th Edge has the set \( D_j \) for \( j \in \mathbb{Z}_N \). For example, each set \( D \) means ‘Temperature’, ‘Humidity’, etc. The problem is how weights \( \omega_{ij} \) and \( v_i \) are updated. We explain Table I for short. Eq. (1) is rewritten as follows:

\[
h_{1i}(x^l) = \tau \left( \sum_{j=0}^{N} w_{ij}x_j^l \right) = \tau \left( \sum_{j=0}^{N} s_{ij}^l \right)
\]  

(17)

where \( s_{ij}^l = w_{ij}x_j^l \) for \( i \in \mathbb{Z}_M, j \in \mathbb{Z}_N^l \) and \( l \in \mathbb{Z}_L \). At Step 1, the term \( s_{ij}^l \) for \( i \in \mathbb{Z}_M, j \in \mathbb{Z}_N^l \) and \( l \in \mathbb{Z}_L \) is computed using the set \( D_j \) and \( w_{ij} \) at Edge \( j \). The result for each Edge is sent to Edge 0. At Step 2, \( h(x^l) \) is computed. Further, \( \delta_j(x^l) \) and \( \delta_{1i}(x^l) \) are calculated and \( \delta_{1i}(x^l) \) is sent to each Edge. Furthermore, from the results, the weights \( v_i \) and \( w_{i0} \) for \( i \in \mathbb{Z}_M \) are updated. At Step 3, the weight \( w_{ij} \) is updated using \( s_{ij} \); \( s_{ij} = w_{ij}x_j^l \) is calculated and sent to Edge 0 for the next learning step. Completion of the algorithm is checked at Step 4.

At Step 2 in Table I, Edge 0 sends all \( \delta_{1i}(x^l) \)'s to Edge \( j \). The number of data sent from Edge 0 to Edge \( j \) is \( M \times L \) and \( \Delta \omega_{ij} \) is computed using them.

On the other hand, let us define \( p_{ij} \) using \( \delta_{1i}(x^l) \) at Step 2 as follows:
Firstly, secure NG learning proposed in Ref. [8] on vertically partitioned data with edge system is
3.2 Simplified NG learning for vertically partitioned data with IoT

Table I. The BP learning method on vertically partitioned data.

| Step       | Edge 0                                      | Edge j                                      |
|------------|---------------------------------------------|---------------------------------------------|
| Initial condition | $v = (v_0, v_1, \cdots, v_M)$ and $w_{i0}$ for $i \in Z_M$ are selected randomly. $D_{out}$, $\alpha_1$, $T_{max}$ and $\theta$ are given. Set $t = 1.$ | The set $D_j$ is given. The weight $w_{ij}$ for $i \in Z_M$ is selected randomly. The constant $\alpha_2$ is given. |
| Step 1     | Calculate $s_{ij}^l = w_{ij}x_j^l$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. | Calculate $s_{ij}^l = w_{ij}x_j^l$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. |
| Step 2     | Calculate $h_{1i}(x^l) = \tau(w_{i0} + \sum_{j=1}^{N} s_{ij}^l)$ and $h(x^l) = \tau(v_0 + \sum_{i=1}^{M} v_i h_{1i}(x^l))$ for $l \in Z_L.$
Calculate $\delta_2(x^l)$ and $\delta_1(x^l)$ for $i \in Z_M$ and $\Delta v_i$ of Eq. (9) and $v_i \leftarrow v_i + \Delta v_i$. $\delta_1(x^l)$ for $i \in Z_M$ and $l \in Z_L$ is sent to each Edge.
Calculate $\Delta w_{i0} = -\sum_{l \in Z_L} \alpha_1 \delta_1(x^l)$ and $w_{i0} \leftarrow w_{i0} + \Delta w_{i0}$ for $i \in Z_M.$ | Calculate $\Delta w_{ij} = -\sum_{l \in Z_L} \alpha_2 \delta_1(x^l)x_j^l$ and $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$. Calculate $s_{ij}^l = w_{ij}x_j^l$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. |
| Step 4     | Calculate $h_{1i}(x^l) = \tau(w_{i0} + \sum_{j=1}^{N} s_{ij}^l)$ and $h(x^l) = \tau(v_0 + \sum_{i=1}^{M} v_i h_{1i}(x^l))$ for $l \in Z_L$ and $E(t) = \frac{1}{2L} \sum_{l=1}^{L} (h(x^l) - d(x^l))^2.$
If $E(t) < \theta$ or $t \geq T_{max}$ then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$ |

\[ p_{ij} = \sum_{l=1}^{L} \delta_{1i}(x^l)s_{ij}^l \quad \text{for} \quad i \in Z_M \quad \text{and} \quad j \in Z_N. \quad (18) \]

By introducing $p_{ij}$, the amount of data transfer from Edge 0 to Edge $j$ to $1/L$ is reduced compared with Table I. By using it, let us propose a simplified algorithm shown in Table II.

As the initial condition, Edge 0 has the weight $v$ and each Edge $j$ has the set $D_j$ and the weight $w_{ij}$ for $i \in Z_M$. At Step 1, Edge $j$ calculates the weighted input $s_{ij}^l$ for $i \in Z_M$ and $l \in Z_L$ and $l$ is sent to Edge 0. At Step 2 of Edge 0, $h_{1i}(x^l) = \tau\left(\sum_{j=0}^{N} s_{ij}^l\right)$ for $i \in Z_M$ is calculated using $s_{ij}^l$ for $j \in Z_M$ and $h(x^l)$ for $l \in Z_L$ is calculated using $h_{1i}(x^l)$. Further, the errors $\delta_2(x^l)$ and $\delta_1(x^l)$ for each data $x^l \in D$ are calculated and the weight $v$ and $w_{i0}$ for $i \in Z_M$ are updated based on Eqs. (8) and (9). Furthermore, $\Delta p_{ij}$ is computed based on Eq. (18) and sent to each Edge. At Step 3 of Edge $j$, $w_{ij}$ is updated using $\Delta p_{ij}$. Further, $s_{ij}^l$ for $i \in Z_M$ and $l \in Z_L$ is calculated and is sent to Edge 0. At Step 4 of Edge 0, $h_{1i}(x^l)$ and $h(x^l)$ for $l \in Z_L$ are calculated as the same computation as Step 2. Further, the difference between output $h(x)$ and the desired output $d(x)$ is calculated and evaluated. If the error $E(t)$ is sufficiently small, then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$.

Remark that each element of $x$ belongs to different Edge. The relation of the numbers of transfer data between Edge 0 and Edge $j$ for Tables I and II is shown in Section 3.3.

### 3.2 Simplified NG learning for vertically partitioned data with IoT

Firstly, secure NG learning proposed in Ref. [8] on vertically partitioned data with edge system is presented (See Table III). The problem of NG is how the weight $w_i$ is updated. Let $D_j = \{x_j^1, \cdots, x_j^L\}$ for $j \in Z_N$. The $j$-th element of Eq. (13) in batch learning is presented as follows:

\[ \Delta w_{ij} = \sum_{l \in Z_L} \alpha h_\lambda(k_l(x^l_i, w_i))(w_{ij} - x_j^l) \quad (19) \]
The constant $\lambda$ for $T_{\max}$ and $\theta$ are given. Set $t = 1$. $s_{l0}^t = w_{i0}$

Step 1
Calculate $h_{ii}(x^l) = \tau (\sum_{j=0}^N s_{ij})$ and $\bar{h}(x^l) = \tau (\sum_{j=0}^M e_i h_{ii}(x^l))$ for $l \in Z_L$. Calculate $p_{ij} = \sum_{l=1}^L \delta_{ii}(x^l)s_{lj}^t$ for $i \in Z_M$ and $j \in Z_N$ and send them to each Edge. Calculate $\Delta w_{i0} = - \sum_{l \in Z_L} \alpha_1 \delta_{ii}(x^l)$ and $\Delta v_l$ of Eq. (9) and $v_l \leftarrow v_l + \Delta v_l$ and $w_{i0} \leftarrow w_{i0} + \Delta w_{i0}$ for $i \in Z_M$.

Step 2
Calculate $h_{ii}(x^l) = \tau (\sum_{j=0}^N s_{ij})$ and $\bar{h}(x^l) = \tau (\sum_{j=0}^M e_i h_{ii}(x^l))$ for $l \in Z_L$ and $E(t) = \frac{1}{2} \sum_{l=1}^L (h(x^l) - \bar{h}(x^l))^2$. If $E(t) < \theta$ or $t \geq T_{\max}$ then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$ and $s_{l0}^t = w_{i0}$

Step 4
Calculate $h_{ii}(x^l) = \tau (\sum_{j=0}^N s_{ij})$ and $\bar{h}(x^l) = \tau (\sum_{j=0}^M e_i h_{ii}(x^l))$ for $l \in Z_L$ and $E(t) = \frac{1}{2} \sum_{l=1}^L (h(x^l) - \bar{h}(x^l))^2$. If $E(t) < \theta$ or $t \geq T_{\max}$ then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$ and $s_{l0}^t = w_{i0}$

Table II. The simplified BP learning method on vertically partitioned data.

| Edge 0 | Edge j |
|-------|--------|
| **Initial condition** | The set $D_j$ is given. The weight $w_{i0}$ for $i \in Z_M$ are selected randomly. $D_{out}$, $\alpha_1$, $T_{\max}$ and $\theta$ are given. Set $t = 1$. $s_{l0}^t = w_{i0}$ |
| **Step 1** | Calculate $s_{ij}^t = w_{ij}x_j^t$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. |
| **Step 2** | The constant $\alpha_2$ is given. |

Table III. The NG learning method on vertically partitioned data [8].

| Edge 0 | Edge j |
|-------|--------|
| **Initial condition** | The set $D_j$ is given. The weight $w_{i0}$ for $i \in Z_r$ is selected randomly. The constant $\alpha$ is given. |
| **Step 1** | Calculate $q_{ij}^t = w_{ij} - x_j^t$ for $l \in Z_L$ and $i \in Z_r$ and send them to Edge 0. |
| **Step 2** | The rank $k_i(x^l, w_i)$ is determined using $d(x^l, w_i)$ and send them to each Edge. |
| **Step 3** | Calculate $\Delta w_{ij} = \alpha \sum_{l \in Z_L} h_\lambda(k_i(x^l, w_i))q_{ij}^t$ for $i \in Z_r$ and $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$ for $i \in Z_r$. |
| **Step 4** | Calculate $q_{ij}^t = w_{ij} - x_j^t$ for $l \in Z_L$ and $i \in Z_r$ and send them to Edge 0. |

where $h_\lambda(k(x, w)) = \exp(-k(x, w)/\lambda)$.

We explain Table III for short. As initial conditions, Edge $j$ stores the set $D_j = \{x_j^1, \ldots, x_j^L\}$ and the weight $w_{i,j}$. At Step 1, the difference $q_{ij}^t = w_{ij} - x_j^t$ is calculated and sent to Edge 0. At Step 2, the rank $k_i(x^l, w_i)$ is calculated using $q_{ij}^t$ and send them to each Edge. At Step 3, the update amount $\Delta w_{ij}$ is calculated and the weight $w_{ij}$ is updated using $\Delta w_{ij}$. Further, $q_{ij}^t$ is updated and sent to Edge 0 for the next processing. At Step 4, completion of the algorithm is checked.

In this method, the number of data for Edge $j$ is $M \times L$. In order to simplify the method, the learning method of Table IV is proposed. The key idea is to compute the update amount $\delta_{ij}$ in Edge
Table IV. The simplified NG learning method on vertically partitioned data.

| Step | Edge 0 | Edge j |
|------|--------|--------|
| Initial condition | \(\alpha \) and \(T_{\text{max}}\) are given. Set \(t = 1\). | The set \(D_j\) is given. The weight \(w_{ij}\) for \(i \in Z_r\) is selected randomly. The constant \(\alpha\) is given. |
| Step 1 | Calculate \(d(x^t, w_i) = \sqrt{\sum_{j=1}^{n}(q_{ij})^2}\) for \(l \in Z_L\). The rank \(k_i(x^t, w_i)\) for \(l \in Z_L\) and \(i \in Z_r\) is determined using \(d(x^t, w_i)\). Calculate \(\delta_{ij} = \alpha \sum_{l \in Z_L} h_x(k_i(x^t, w_i))(w_{ij} - x^t_j)\) for \(i \in Z_r\) and \(j \in Z_N\) is calculated in Edge 0 and sent to each Edge. | Calculate \(q_{ij}^t = w_{ij} - x^t_j\) for \(l \in Z_L\) and \(i \in Z_r\) and send them to Edge 0. |
| Step 2 | | Calculate \(w_{ij} \leftarrow w_{ij} + \delta_{ij}\) for \(i \in Z_r\). Calculate \(q_{ij}^t = w_{ij} - x^t_j\) for \(l \in Z_L\) and \(i \in Z_r\) and send them to Edge 0. |
| Step 3 | | |
| Step 4 | If \(t \geq T_{\text{max}}\) then the algorithm terminates else go to Step 2 with \(t \leftarrow t + 1\). | |

0 instead of computing \(\Delta w_{ij}\) in each Edge.

We explain Table IV. At Step 1 of Edge \(j\), the difference \(q_{ij}^t\) between the weight \(w_{ij}\) for \(i \in Z_r\) and the \(l\)-th input data \(x^t_j\) is calculated. At Step 2 of Edge 0, the distance \(d(x^t, w_i)\) of input data \(x^t\) and the weight \(w_i\) for \(i \in Z_r\) is calculated, the rank \(k_i(x^t, w_i)\) for \(i \in Z_r\) is determined and sent to each Edge. Further, the update amount \(\delta_{ij} = \alpha \sum_{l \in Z_L} h_x(k_i(x^t, w_i))(w_{ij} - x^t_j)\) for \(i \in Z_r\) and \(j \in Z_N\) is calculated in Edge 0 and sent to each Edge. At Step 3 of Edge \(j\), \(w_{ij} \leftarrow w_{ij} + \delta_{ij}\) for \(i \in Z_r\) are calculated. Further, \(s_{ij}\) is calculated and sent to Edge 0. At Step 4 of Edge 0, it is checked if the number of epochs is sufficient.

Remark that each element of \(x\) belongs to different Edge. The relation of the numbers of transfer data between Edge 0 and Edge \(j\) for Tables III and IV is shown in Section 3.3.

3.3 The relation between the proposed and conventional methods

Let us summarize what is new in the proposed methods compared to the conventional methods. First, let us explain the case of BP. Figure 4(a) shows the initial assignment of data and parameters and the relation of data transfer for algorithms of Tables I and II. Output \(d(x^t)\), threshold value \(w^0_i\) of the first layer and weight \(v_i\) of the second layer are assigned to Edge 0, and \(j\)-th components \(x^t_j\) and \(w_{ij}\) of data and parameters are assigned to Edge \(j\) for \(j \in Z_N\). First, each Edge \(j\) calculates the product \(s^t_{ij} = w_{ij}x^t_j\) and sends them to Edge 0. The data \(x^t_j\) itself does not go out from Edge \(j\). Edge 0 receives the product \(s^t_{ij}\) of each Edge and calculates the update amounts of the weight \(w_{ij}\). In this case, the following methods for calculating the update amount of \(w_{ij}\) can be considered (See Fig. 4(a)):  
1) At Edge 0, \(M \times L\) pieces of data \(\delta_{ij}(x^t)\) are calculated and sent to each Edge \(j\). At Edge \(j\), \(\Delta w_{ij}\) is calculated based on Eq. (5), and \(w_{ij}\) is updated.
2) At Edge 0, \(M\) pieces of data \(p_{ij}\) of Eq. (18) are calculated and sent to Edge \(j\). At Edge \(j\), \(p_{ij}/w_{ij}\) is calculated, and \(w_{ij}\) is updated.

In the previous paper, the algorithm based on the above method 1) was proposed as Table I [7]. In this paper, the algorithm based on the above method 2) is proposed as Table II. In this case, the numbers of parameters to be transferred from Edge 0 to Edge \(j\) can be estimated as shown in Table V(a). Parameters \(w_{ij}\) and \(x^t_j\) for \(i \in Z_L\) are assigned to Edge \(j\) and \(\delta_{ij}(x^t)\) in the method 1) or \(p_{ij}\) in the method 2) is assigned to Edge 0. That is, the transfer of the update amount can be reduced to \(1/L\) by sending \(p_{ij}\) instead of \(\delta_{ij}(x^t)\).

Similarly, let us consider the case of NG. Figure 4(b) shows the initial assignment of data and
Fig. 4. The relation about data transfer between methods 1) and 2): In the figures, the content of each variable and data on the arrow mean the initial assignment of parameters and the transferred data between Edge 0 and Edge \(j\), respectively.

(a) For BP: the amounts of data transferred by method 1) using \(\delta_{ji}(x^l)\) or method 2) using \(p_{ij}\), where each set means \(\{d(x^l)|l\in Z_L\}\), \(\{w_{i0}, v_i|l\in Z_M\}\), \(\{x^l_j|l\in Z_L\}\) and \(\{w_{ij}|l\in Z_M\}\) for \(j\in Z_N\).

(b) For NG: the amounts of data transferred by method 1) using \(k_i(x^l, w_i)\) or method 2) using \(\delta_{ij}\), where each set means \(\{x^l_j|l\in Z_L\}\) and \(\{w_{ij}|l\in Z_M\}\) for \(j\in Z_N\).

**Table V.** The numbers of parameters used in each Edge.

(a) BP learning method for Tables I and II.

| data       | Previous (Table I [7]) | Proposed (Table II) |
|------------|-------------------------|---------------------|
| \(x^l_j\)  | \(L\)                   | \(L\)               |
| \(w_{ij}\) | \(M\)                   | \(M\)               |
| \(\delta_{ij}(x^l)\) | \(M \times L\) | \(M \times L\) |
| \(p_{ij}\) |                        | \(M\)               |
| Total      | \(L \times M + L + M\) | \(L + 2M\)          |

(b) NG learning method for Tables III and IV.

| data       | Previous (Table III [8]) | Proposed (Table IV) |
|------------|--------------------------|---------------------|
| \(x^l_j\)  | \(L\)                    | \(L\)               |
| \(w_{ij}\) |                          | \(r\)               |
| \(k_i(x^l, w_i)\) | \(r \times L\) | \(r \times L\) |
| \(\delta_{ij}\) |                        | \(r\)               |
| Total      | \(L \times r + L + r\)  | \(L + 2r\)          |

Parameters of methods and the relation of data transfer for Tables III and IV. In Table III, \(k_i(x^l, w_i)\)'s are calculated at Edge 0, sent to Edge \(j\) and parameters \(w_{ij}\)'s are updated at Edge \(j\). In the proposed method of Table IV, \(\delta_{ij}\)'s instead of \(\Delta w_{ij}\) are calculated at Edge 0, sent to each Edge and parameters \(w_{ij}\)'s are updated. This relationship is shown in Fig. 4(b). The amounts of parameters transferred from Edge 0 to Edge \(j\) are \(r \times L\) pieces of \(k_i(x^l, w_i)\) in Table III and \(r\) pieces of \(\delta_{ij}\) in Table IV, respectively (See Table V(b)). That is, the amount of transferred parameters can be reduced to \(1/L\) by sending \(\delta_{ij}\) instead of \(k_i(x^l, w_i)\).

**Table VI.** The dataset for pattern classification.

|                | Iris | Wine | Sonar | BCW | Spam |
|----------------|------|------|-------|-----|------|
| # data         | 150  | 178  | 208   | 683 | 4601 |
| # input        | 4    | 13   | 60    | 9   | 57   |
| # class        | 3    | 3    | 2     | 2   | 2    |

421
4. Numerical simulations for the proposed algorithms

4.1 Numerical simulations for BP methods

In this section, numerical simulation for pattern classification using benchmark problems of Iris, Wine, Sonar, BCW and Spam in the UCI database [12] as shown in Table VI is performed. The BP (online) and Batch methods mean the conventional BP in online and batch using one server, respectively. Previous and Proposed mean the methods presented in Tables I and II, respectively. The number of second layers as $M$ is 10 and the maximum epoch as $T_{\text{max}}$ is 100000 for all cases. The threshold $\theta$ is 0.01 for Iris and Wine and 0.02 for Sonar and BCW. In Table VI, #data means the number of data. In this simulation, 5-fold cross validation as an evaluation method is used. Table VII shows the results of numerical simulations. In each box of Table VII, Training and Test mean the rate (%) of misclassified data for training and test, respectively. The misclassification rate means the ratio (%) of the misclassified data to all data. Each value is average from twenty trials.

|                | Iris | Wine | Sonar | BCW | Spam |
|----------------|------|------|-------|-----|------|
| **BP (online)**|      |      |       |     |      |
| Training       | 3.1  | 6.00 | 1.60  | 3.95| 9.9  |
| Test           | 3.3  | 6.00 | 16.81 | 3.12| 10.2 |
| **Batch**      |      |      |       |     |      |
| Training       | 3.6  | 1.87 | 1.13  | 2.33| 3.9  |
| Test           | 3.9  | 5.94 | 16.48 | 2.98| 6.8  |
| **Previous**   |      |      |       |     |      |
| ($N = \#\text{input}$) Training | 3.5  | 1.9  | 1.2   | 2.3 | 3.9  |
| ($N = \#\text{input}$) Test       | 3.7  | 5.6  | 17.0  | 2.8 | 6.8  |
| **Proposed**   |      |      |       |     |      |
| ($N = \#\text{input}$) Training | 3.6  | 1.91 | 1.25  | 2.33| 3.9  |
| ($N = \#\text{input}$) Test       | 3.7  | 5.83 | 17.10 | 2.82| 6.7  |

All results are almost the same, with some variations. That is, it is shown that BP learning using four models can achieve the same degree of accuracy.

The numbers of data transfer are 1200, 1420, 1660 and 5460 for Iris, Wine, Sonar and BCW in the case of Previous in Table VII, respectively. On the other hand, the numbers of data transfer are 10 for Iris, Wine, Sonar and BCW in the case of Proposed in Table VII, respectively.

4.2 Numerical simulations for NG methods

Five real-world datasets of Iris, Wine, Sonar, BCW and Spam from UCI machine learning repository have been considered [12]. The number of reference vectors is same as \#class in Table VI. The maximum epochs as $T_{\text{max}}$ are 15000 for Iris, 18000 for Wine, 21000 for Sonar, 70000 for BCW and 45000 for Spam and NG as vector quantization techniques is used in the simulation. The problem is how each dataset is approximated by reference vectors.

Table VIII shows the results of the misclassification rates for simulations. The misclassification rate means the ratio (%) of the misclassified data to all data. The notation in Table VIII is same as in Table VII. Each result of simulations is the average from twenty trials. The result shows that the accuracy between the conventional and the proposed methods is almost the same.

We explain the number of data transfer same as the section 4.1. The numbers of data transfer are 450, 534, 416 and 1366 for Iris, Wine, Sonar and BCW in the case of Previous in Table VIII, respectively. On the other hand, the numbers of data transfer are 3 for Iris and Wine and 2 for Sonar and BCW in the case of Proposed in Table VIII, respectively.

A real edge computing system is expected to consist of many edge nodes. As show Tables VII and VIII, even if the number of inputs is increased from 4 to 60, the same approximation accuracy as the conventional method is nominated. That is, it does not seem to depend much in the number of servers.
### Table VIII. Result for clustering problems by using the simplified NG learning.

|                | Iris | Wine | Sonar | BCW | Spam |
|----------------|------|------|-------|-----|------|
| **NG (online)**| 4.2  | 6.9  | 45.3  | 3.6 | 23.8 |
| **Batch**      | 4.0  | 6.7  | 45.1  | 3.5 | 20.4 |
| **Previous**   | 4.0  | 6.9  | 45.2  | 3.5 | 21.3 |
| \((N = \#input)\) | 4.0  | 6.8  | 45.2  | 3.5 | 20.4 |

### 5. Conclusion

In this paper, simplified BP and NG learning on vertically partitioned data with edge system were proposed. The learning process consists of calculation and data transfer. In the conventional method using vertically partitioned data on edge systems, the calculation part is distributed to each server. This paper proposed simplified learning methods that eliminate wasteful data transfer compared to the methods in the previous paper, and its effectiveness is shown. In Section 2, edge computing systems and a secure data sharing mechanism used in this paper were explained. Further, the conventional BP and NG methods were introduced. In Section 3, simplified BP and NG learning methods on vertically partitioned data with edge system were proposed. In section 4, numerical simulation was performed to show the performance of the proposed methods.

In the future, other application for SDM and other methods for dividing data will be considered.

### References

[1] M.G.S. Murshed, et al., “Machine learning at the network edge: A survey,” https://arxiv.org/pdf/1908.00080.pdf, 2020 (accessed 11 March 2020).

[2] J. Chan and X. Ran, “Deep learning with edge computing: A review,” *Proc. of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[3] C.C. Aggarwal and P.S. Yu, “Privacy-preserving data mining: Models and algorithms,” ISBN 978-0-387-70991-8, *Springer-Verlag*, 2009.

[4] H. Miyajima, H. Miyajima, and N. Shiratori, “Fast and secure edge-computing algorithms for classification problems,” *IAENG International Journal of Computer Science*, vol. 46, no. 4, pp. 512–517, 2019.

[5] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyaniishi, S. Kitagami, and N. Shiratori, “New privacy preserving back propagation learning for secure multiparty computation,” *IAENG International Journal of Computer Science*, vol. 43, No. 3, pp. 270–276, 2016.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Article 12, 2019.

[7] H. Miyajima, H. Miyajima, and N. Shiratori, “Fast and secure back-propagation learning using vertically partitioned data with IoT,” *CANDAR 2019: The Seventh International Symposium on Computing and Networking, Nagasaki, November*, pp. 450–454, 2019.

[8] H. Miyajima, N. Shigei, H. Miyajima, and N. Shiratori, “Fast and secure learning algorithms for classification problems using vertically partitioned data with IoT,” *International Symposium on Artificial Life and Robotics 2020*, pp. 321–326, 2020.

[9] M.M. Gupta, L. Jin, and N. Homma, “Static and dynamic neural networks,” *IEEE Press*, 2003.

[10] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten, “Neural Gas network for vector quantization and its application to time-series prediction” *IEEE Trans. Neural Network*, vol. 4, no. 4, pp. 558–569, 1993.

[11] C. Gentry, “Fully homomorphic encryption using ideal lattices,” *Proc. 41st Annual ACM Symposium on Theory of Computing*, STOC 2009, pp. 169–178, 2009.

[12] UCI Repository of Machine Learning Databases and Domain Theories, ftp://ftp.ics.uci.edu/pub/machinelearning-Databases.