Proceedings of the International MultiConference of Engineers and Computer Scientists 2021
IMECS 2021, October 20-22, 2021, Hong Kong

Federated Learning with Divided Data for BP

Hirofumi Miyajima, Noritaka Shigei, Hiromi Miyajima, and Norio Shiratori,

***Abstract*—The edge computing system is known as a method for improving the conventional cloud system. Some machine learning methods for maintaining data security are proposed for the edge computing system. In Federated Learning (FL) using Horizontally Partitioned Data (HPD) or Vertically Partitioned Data (VPD), the method of sharing data and learning is used. On the other hand, we proposed a federated Back Propagation (BP) learning with divided data in the previous paper. In this paper, we generalize this method, propose a method of dividing data for each epoch in learning and show its effectiveness in numerical simulations.**

***Index Terms*—Federated Learning, Edge Computing, Back Propagation Method, Divided Data, Horizontally Partitioned Data.**

I. Introduction

CLOUD computing is one of the basic technologies that support Information and Communication Technology and is used in various fields. With the transition to the Internet of Things (IoT), the number of servers (things) connected to the cloud system is increasing exponentially. Therefore, the load on the servers increases dramatically and the processing power of the cloud system is significantly reduced. The edge computing system is a method for complimenting the conventional cloud system [1], [2], [3]. This is a model in which the main servers located far away in the conventional system are replaced with local servers located near terminals or things. It means a shortcut for the tasks for main servers. However, the processing power of edge system servers is less powerful than one of the cloud servers. Furthermore, there is also the issue of the security of the data used there. Therefore, the problem is how to realize complicated calculations by combining multiple servers with low capacity while maintaining the confidentiality of data. As a solution for the problem, FL using HPD or VPD is proposed, but learning data are not so secured because each data is used as it is [3], [4], [5]. On the other hand, we proposed a secure learning method [6]. In the method, each of the learning data and parameters is divided into multiple pieces randomly and they are used as learning data, so data seems to be secured. It is called the securely distributed computation with divided learning data (SDCD). However, in this method, divided data are used as they are until the learning is completed. To improve the safety of learning data, it is desirable to divide data for each epoch in learning. In this paper, we propose a SDCD of dividing data for each epoch in learning. In particular, the generalized BP learning with divided data is proposed. The accuracy of the proposed SDCD and the conventional methods is compared by numerical simulations.

Hirofumi Miyajima is an Associate Professo of Nagasaki University, 1-14 Bunkyomachi, Nagasaki city, Nagasaki, Japan (e-mail: miyajima@nagasaki-u.ac.jp)

Noritaka Shigei is an Associate Professo of Kagoshima University, 1-21-24, Korimoto, Kagoshima, Japan (e-mail: shigei@eee.kagoshima-u.ac.jp).

Hiromi Miyajima is a Professor Emeritus of Kagoshima University, 1-21-24, Korimoto, Kagoshima, Japan (e-mail: k2356323@kadai.jp).

Norio Shiratori is a Professor of Chuo University, 1-13-27, Kasuga, Bunkyoku, Tokyo, Japan (e-mail: norio@riec.tohoku.ac.jp).

II. Preliminaries

A. Secure computation and a configuration for the edge system

Regarding cloud and edge 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 [7], 2) Homomorphic encryption [8], 3) Federated learning. Studies on 1) and 2) are methods based on encryption, and 3) is a method in which dataset is partitioned and learning is performed by distributed computing. Method 3) shares the dataset into multiple subsets by using HPD or VPD and performs machine learning using distributed system so that the dataset is not concentrated in one sever. Method 3) is also considered to be suitable for application to edge systems. The proposed method in this paper uses secure divided data and FL. In the following, we will introduce how to realize machine learning by distributed processing while preserving security using divided data.

Let us explain a system for the proposed method. Figure 1 shows the system with L terminals and $Q+1$ servers. Let x and f be a (scalar) data and a function. In Ref.[2], Server 0 is also called the aggregator. Each data x is divided into some pieces and each of them is sent to each server.

First, any data x in each terminal is divided into Q pieces randomly as $x = \sum_{q=1}^{Q} x^{(q)}$. A piece $x^{(q)}$ is sent to Server q. Each function $f_q(x^{(q)})$ is calculated in Server q and sent to Server 0, where $f_q(\cdot)$ means a function in Server q. In Server 0, $f_1(x^{(1)}),\cdots,f_q(x^{(q)}),\cdots$ and $f_Q(x^{(Q)})$ are aggregated and $f(x) = \odot_{q=1}^{Q} f_q(x^{(q)})$ is calculated, where $\odot$ means an integrated calculation. If the calculation result is not obtained in one process, multiple processes are repeated.

The problem is how to determine the calculation process $f_q(x^{(q)})$, where $f(x) = \odot_{q=1}^{Q} f_q(x^{(q)})$.

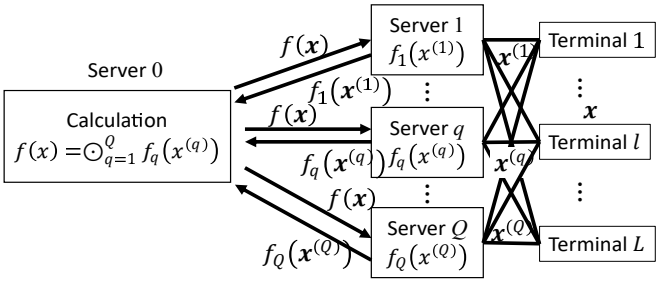

Fig. 1. An example of SDCD : The data given from each terminal is divided into Q pieces and sent to each server. At each server, $f_q(x^{(q)})$ is calculated and sent to Server 0. On Server 0, the partial calculation $f_q(x^{(q)})$ is integrated to obtain the result $f(x)$. This result is sent to each server and used every time $f_q(x^{(q)})$ is updated.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
EXAMPLE OF DATA DIVISION FOR THE PROPOSED METHOD (ADDITION)

	data a	data b	addition
Server 1	$a^{(1)}$ =2	$b^{(1)}$ =5	7
Server 2	$a^{(2)}$ =4	$b^{(2)}$ =-3	1
Server 3	$a^{(3)}$ =-3	$b^{(3)}$ =2	-1
addition	3	4	7

TABLE II
EXAMPLE OF DATA DIVISION FOR THE PROPOSED METHOD (MULTIPLICATION)

	data a	data b	multiplication
Server 1	$A^{(1)}$ =-1	$B^{(1)}$ =-2	2
Server 2	$A^{(2)}$ =3	$B^{(2)}$ =-1	-3
Server 3	$A^{(3)}$ =-1	$B^{(3)}$ =2	-2
multiplication	3	4	12

B. Data division for the proposed method

Let us explain data representation used in the proposed method [9]. Let a and b be two positive integers and the number of servers is 3. First, two integers a and b are divided into three real numbers. Let $a = a^{(1)} + a^{(2)} + a^{(3)}$ and $b = b^{(1)} + b^{(2)} + b^{(3)}$ as addition form and $a = A^{(1)}A^{(2)}A^{(3)}$ and $b = B^{(1)}B^{(2)}B^{(3)}$ as the multiplication form. Then the following results hold:
1)$a + b = (a^{(1)} + b^{(1)}) + (a^{(2)} + b^{(2)}) + (a^{(3)} + b^{(3)})$
2)$a - b = (a^{(1)} - b^{(1)}) + (a^{(2)} - b^{(2)}) + (a^{(3)} - b^{(3)})$
3)$ab = (A^{(1)}B^{(1)})(A^{(2)}B^{(2)})(A^{(3)}B^{(3)})$
4)$a/b = (A^{(1)}/B^{(1)})(A^{(2)}/B^{(2)})(A^{(3)}/B^{(3)})$

That is, four basic operations of arithmetic (addition, subtraction, multiplication, and division) hold as the integration of the result computed independently by each server. In this case, each server can not know the original data a and b.

For example, let $a = 3$ and $b = 4$. a and b are divided as $3 = 2+4+(-3) = (-1)\times 3\times(-1)$ and $4 = 5+(-3)+2 = (-2)\times(-1)\times 2$. In this case, $a^{(1)} = 2$, $a^{(2)} = 4$, $a^{(3)} = -3$, $A^{(1)} = -1$, $A^{(2)} = 3$, $A^{(3)} = -1$, $b^{(1)} = 5$, $b^{(2)} = -3$, $b^{(3)} = 2$, $B^{(1)} = -2$, $B^{(2)} = -1$ and $B^{(3)} = 2$, respectively. $a + b$ can be calculated as follows (See Table I).

$$\begin{aligned} a+b &= (a^{(1)} + b^{(1)}) + (a^{(2)} + b^{(2)}) + (a^{(3)} + b^{(3)}) \\ &= (2+5) + (4+(-3)) + ((-3)+2) = 7 \end{aligned}$$

$a\times b$ can be calculated as follows (See Table II).

$$\begin{aligned} a\times b &= (A^{(1)}\times B^{(1)})(A^{(2)}\times B^{(2)})(A^{(3)}\times B^{(3)}) \\ &= ((-1)\times(-2))(3\times(-1))((-1)\times 2) = 12 \end{aligned}$$

C. Neural Network and BP method

In this section, let us explain three layered Neural Network (NN) and BP method using Figure 2 without loss of generality [10]. For any positive integer i, let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$. $\boldsymbol{h} : J_{in}^n \to J_{out}^R$ is determined for each $\boldsymbol{x}\in J_{in}^n$ as follows : $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \cdots, h_R(\boldsymbol{x}))$, where $J_{in} = [0, 1]$ or $[-1, 1]$, $J_{out} = \{0, 1\}$. In this case, weights are determined by using the set of learning data $X = \{(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l))|\boldsymbol{x}^l\in J_{in}^n, \boldsymbol{d}(\boldsymbol{x}^l)\in J_{out}^R, l\in Z_L\}$, where $\boldsymbol{d}(\boldsymbol{x}^l) = \big(d_1(\boldsymbol{x}^l), \ldots, d_R(\boldsymbol{x}^l)\big)$ is the desired output for the input data $\boldsymbol{x}^l$.

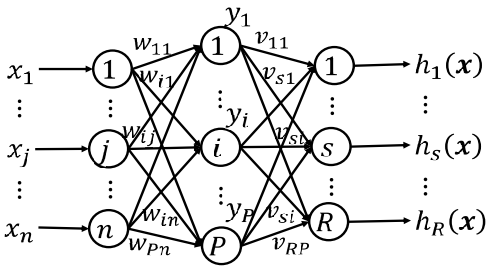

Fig. 2. An example of three layered Neural Network

Let two sets of weights be denoted by $W = \{w_{ij}|i\in Z_P, j\in Z_n^*\}$, $V = \{v_{si}|s\in Z_R, i\in Z_P^*\}$. In this case, an output of NN is calculated as follows :

$$y_i(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{j=0}^n w_{ij}x_j\right)\right)} \tag{1}$$

$$h_s(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{i=0}^P v_{si}y_i(\boldsymbol{x})\right)\right)} \tag{2}$$

where $x_0 = 1$ and $y_0 = 1$.

The evaluation function is defined as follow:

$$E(X, W, V) = \frac{1}{2L}\sum_{l=1}^{L}\sum_{s=1}^{R}\big(d_s(\boldsymbol{x}^l) - h_s(\boldsymbol{x}^l)\big)^2 \tag{3}$$

Each weight of W and V is determined by using the BP method as follows [10]. In this case, X, T, θ, and α mean the set of learning data, the maximum number of learning time, threshold, and learning rate, respectively.

[**B**P method] BP(X, W, V)

Input : Learning data $X = \{(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l))|\boldsymbol{x}^l\in J_{in}^n, \boldsymbol{d}(\boldsymbol{x}^l)\in J_{out}^R, l\in Z_L\}$

Output : Weights $W = \{w_{ij}|i\in Z_P, j\in Z_n^*\}$ and $V = \{v_{si}|s\in Z_R, i\in Z_P^*\}$

[**S**tep 1]

Initialize W, V and $t\leftarrow 0$.

[**S**tep 2]

Select a learning data $(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l))\in X$ randomly. By using Eqs.(1) and (2), $y_i(\boldsymbol{x}^l)$ and $h_s(\boldsymbol{x}^l)$ are calculated.

[**S**tep 3]

By using the following equations, $w_{ij}\in W$ and $v_{si}\in V$ are updated.

$$\begin{aligned} w_{ij} &\leftarrow w_{ij} + \alpha\sum_{s=1}^{S}(d_s(\boldsymbol{x}^l) - h_s(\boldsymbol{x}^l))(1-h_s(\boldsymbol{x}^l))v_{si} \\ &\qquad \times y_i(\boldsymbol{x}^l)(1-y_i(\boldsymbol{x}^l))x_j^l \qquad (4) \\ v_{si} &\leftarrow v_{si} + \alpha(d_s(\boldsymbol{x}^l) - h_s(\boldsymbol{x}^l))h_s(\boldsymbol{x}^l) \\ &\qquad \times(1-h_s(\boldsymbol{x}^l))y_i(\boldsymbol{x}^l) \qquad (5) \end{aligned}$$

[**S**tep 4]

By using Eq.(3), the evaluation value $E(X, W, V)$ is calculated.

[**S**tep 5]

If $E < \theta$ or $t < T$, the algorithm terminates else go to Step 2 with $t\leftarrow t+1$.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

III. Generalized BP learning for SDCD

A. Data structure for the proposed method

In this section, any learning data $(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l))\in X$ is divided with Q pieces and they are stored in each server as follows [6] :

$$x_j^l = \Pi_{q=1}^Q x_j^{l(q)} \quad (6)$$

$$d_s(\boldsymbol{x}^l) = \sum_{q=1}^Q d_s^{(q)}(\boldsymbol{x}^l) \quad (7)$$

In the same way, weights $w_{ij}\in W$ and $v_{si}\in V$ are divided with Q pieces and they are stored in each server as follows:

$$w_{ij} = \Pi_{q=1}^Q w_{ij}^{(q)} \quad (8)$$

$$v_{si} = \Pi_{q=1}^Q v_{si}^{(q)} \quad (9)$$

In this case, let $W^{(q)} = \{w_{ij}^{(q)} | i\in Z_P, j\in Z_n^*\}$ and $V^{(q)} = \{v_{si}^{(q)} | s\in Z_S, i\in Z_P^*\}$.

Let $\Pi_{q=1}^Q x_0^{(q)} = 1$. An output of NN for the divided data based on Eqs.(6), (7), (8), and (9) is calculated instead of (1) and (2) as follows :

$$y_i(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{j=0}^n \Pi_{q=1}^Q w_{ij}^{k(q)} x_j^{(q)}\right)\right))} \quad (10)$$

Further, y_i is divided as $y_i = \Pi_{q=1}^Q y_i^{(q)} (i\in Z_P^*)$ and $y_i^{(q)}$ for $q\in Z_Q$ is sent to server q. Let $\Pi_{q=1}^Q y_0^{(q)} = 1$. An output $h_s(\boldsymbol{x})$ of NN is calculated in Server 0 as follows :

$$h_s(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{i=0}^P \Pi_{q=1}^Q v_{si}^{k(q)} y_i^{(q)}\right)\right)} \quad (11)$$

The output $h_s(\boldsymbol{x})$ is divided as $h_s(\boldsymbol{x}) = \sum_{q=1}^Q h_s^{(q)}(\boldsymbol{x})$ and $h_s^{(q)}(\boldsymbol{x})$ is sent to Server q.

In this case, Mean Square Error (MSE) for the set of learning data is calculated as follows :

$$E(X) = \frac{1}{2L}\sum_{l=1}^L \sum_{s=1}^R \left(\sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))\right)^2 \quad (12)$$

Based on SDM, each of weights $w_{ij}^{(q)} (i\in Z_P, j\in Z_P^*)$ and $v_{si}^{(q)} (s\in Z_S, i\in Z_P^*)$ is updated instead of Eqs.(4) and (5) as follows [6] :

$$\begin{aligned} w_{ij}^{(q)} = w_{ij}^{(q)} + & \quad \alpha \sum_{s=1}^R \sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))(1-h_s(\boldsymbol{x}^l)) \\ & \times \Pi_{q=1}^Q v_{si}^{(q)} h_s^{(q)}(\boldsymbol{x}^l)(1-h_s(\boldsymbol{x}^l)) \\ & \times (\Pi_{q=1}^Q w_{ij}^{(q)} x_j^{l(q)})/w_{ij}^{(q)} \end{aligned} \quad (13)$$

$$\begin{aligned} v_{si}^{(q)} = v_{si}^{(q)} + \alpha & \sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) h_s(\boldsymbol{x}^l) \\ & \times (1-h_s(\boldsymbol{x}^l))(\Pi_{q=1}^Q y_i^{(q)} v_{si}^{(q)})/v_{si}^{(q)}) \end{aligned} \quad (14)$$

B. Update the divided data for the proposed method

In the previous paper, once each of data x_j^l and $d_s(\boldsymbol{x}^l)$ is divided into $x_j^{l(q)}$ and $d_s^{(q)}(\boldsymbol{x}^l) (q\in Z_Q)$, the same division is used through the learning [6].

In this paper, the generalized learning method is proposed. First, it is shown that the data updated by adding and multiplying the random numbers to the divided data in step u, becomes the same data in step $u+1$.

Let $x_j^{l(q)}(u) (j\in Z_n, l\in Z_L, q\in Z_Q)$ be a divided input data at step u. Assume that input data x_j^l is defined as follows:

$$x_j^l = \Pi_{q=1}^Q x_j^{l(q)}(u) \quad (15)$$

At the learning step u, each $O_1^{(q)}(u) (q\in Z_Q)$ is selected randomly satisfying the following equation :

$$\Pi_{q=1}^Q O_1^{(q)}(u) = 1 \quad (16)$$

In this case,

$$\begin{aligned} x_j^l &= x_j^l \times 1 \\ &= \Pi_{q=1}^Q O_1^{(q)} x_j^{l(q)}(u) \end{aligned} \quad (17)$$

By using Eq.(17), $x_j^{l(q)}(u)$ is updated as follows :

$$x_j^{l(q)}(u+1) = O_1^{(q)}(u) \times x_j^{l(q)}(u) \quad (18)$$

In this case, by using Eq.(17), the following equation is obtained.

$$\Pi_{q=1}^Q x_j^{l(q)}(u+1) = x_j^l \quad (19)$$

Therefore, by using Eq.(18), $x_j^{l(q)}(u+1)$ is different from $x_j^{l(q)}(u)$ and $\Pi_{q=1}^Q x_j^{l(q)}(u+1) = \Pi_{q=1}^Q x_j^{l(q)}(u) = x_j^l$.

The result shows that the divided input data can be changed independently on each server.

Similarly, let $d_s^{(q)}(\boldsymbol{x}^l)(u) (s\in Z_S, l\in Z_L, q\in Z_Q)$ be a divided output data when the learning step is u. Assume that output data $d_s(\boldsymbol{x}^l)$ is divided at step u as follows :

$$d_s(\boldsymbol{x}^l) = \sum_{q=1}^Q d_s^{(q)}(\boldsymbol{x}^l)(u) \quad (20)$$

At step u, let $O_0^{(q)}(u) (q\in Z_Q)$ be selected randomly satisfying the following equation :

$$\sum_{q=1}^Q O_0^{(q)}(u) = 0 \quad (21)$$

In this case,

$$\begin{aligned} d_s(\boldsymbol{x}^l) &= d_s(\boldsymbol{x}^l) + 0 \\ &= \sum_{q=1}^Q \left(d_s^{(q)}(\boldsymbol{x}^l)(u) + O_0^{(q)}(u)\right) \end{aligned} \quad (22)$$

By using the equations, $d_s^{(q)}(\boldsymbol{x}^l)(u)$ are updated as follows:

$$d_s^{(q)}(\boldsymbol{x}^l)(u+1) = O_0^{(q)}(u) + d_s^{(q)}(\boldsymbol{x}^l)(u) \quad (23)$$

In this case, by using Eq.(22), the following equation is obtained.

$$\sum_{q=1}^Q d_s^{(q)}(\boldsymbol{x}^l)(u+1) = d_s(\boldsymbol{x}^l) \quad (24)$$

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE III
BP METHOD FOR SDCD.

	Server 0	Server q ($q \in Z_Q$)
Initialize		Store $\{x_j^{l(q)} \mid l \in Z_L, j \in Z_n\}$ and $\{d_s^{(q)}(\boldsymbol{x}^l) \mid l \in Z_L, s \in Z_R\}$. Initialize $\{w_{ij}^{(q)} \mid i \in Z_P, j \in Z_n^*\}$ and $\{v_{si}^{(q)} \mid s \in Z_R, i \in Z_P^*\}$.
Step 1	$t \leftarrow 0$	
Step 2	Select a number $l \in Z_L$ randomly and send it to each server.	
Step 3		Calculate $w_{ij}^{(q)} x_j^{l(q)} (i \in Z_P, j \in Z_n^*)$and send it to Server 0.
Step 4	Calculate $y_i(\boldsymbol{x}^l)$ by using Eq.(10). Divide $y_i = \Pi_{q=1}^{Q} y_i^{(q)}$. Send $y_i^{(q)} (q \in Z_Q)$ to Server q.	
Step 5		Calculate $v_{si}^{(q)} y_i^{(q)} (s \in Z_R, i \in Z_P^*)$ and send it to Server 0.
Step 6	Calculate $h_s(\boldsymbol{x}^l) (s \in Z_S)$ by using Eq.(11). Divide $h_s(\boldsymbol{x}^l) = \Pi_{q=1}^{Q} h_s^{(q)}(\boldsymbol{x}^l)$. Send $h_s^{(q)}(\boldsymbol{x}^l) (q \in Z_Q)$ to Server q.	
Step 7		Calculate $d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l) (s \in Z_R)$ and send it to Server 0.
Step 8	Calculate $p_{1(ij)} = \sum_{s=1}^{R} \sum_{q=1}^{Q} (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))(1 - h_s^{(q)}(\boldsymbol{x}^l)) v_{si} y_i(\boldsymbol{x}^l)(1 - y_i(\boldsymbol{x}^l)) \times (\Pi_{q=1}^{Q} w_{ij}^{(q)} x_j^{l(q)})$ and $p_{2(si)} = \sum_{q=1}^{Q} (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) h_s(\boldsymbol{x}^l)(1 - h_s(\boldsymbol{x}^l)) (\Pi_{q=1}^{Q} y_i^{(q)} v_{si}^{(q)})$ and send them to each server.	
Step 9		Update $\{w_{ij}^{(q)} \mid i \in Z_P, j \in Z_j^*\}$ and $\{v_{si}^{(q)} \mid s \in Z_R, i \in Z_P^*\}$: $w_{ij}^{(q)} \leftarrow w_{ij}^{(q)} + \alpha p_{1(ij)} / w_{ij}^{(q)}$ $v_{si}^{(q)} \leftarrow v_{si}^{(q)} + \alpha p_{2(i)} / v_{si}^{(q)}$
Step 10	Calculate $E(X, W, V)$ by using Eq.(12).	
Step 11	If $E < \theta$ or $t < T$, algorithm terminates. Otherwise go to Step 2 with $t \leftarrow t + 1$.	

Therefore, by using Eq.(23), $d_s^{(q)}(\boldsymbol{x}^l)(u+1)$ is different from $d_s^{(q)}(\boldsymbol{x}^l)(u)$ and $\sum_{q=1}^{Q} d_s^{(q)}(\boldsymbol{x}^l)(u+1) = \sum_{q=1}^{Q} d_s^{(q)}(\boldsymbol{x}^l)(u) = d_s(\boldsymbol{x}^l)$.

The result shows that the divided output data can be changed independently on each server.

C. The generalized BP method for SDCD

Let us explain the general flow of the algorithm shown in Table IV. The data and weights are divided and stored in each server. The partial calculation is performed on each server, and the output of NN is calculated by the integration on Server 0. The resulting output is divided and sent to each server. Each server calculates the error between the sent data and the output that the server has, and sends it to Server 0. Server 0 integrates these data, calculates the updated amount of weight, and sends it to each server. Each server updates the weights using it. If the learning epoch is a multiple of T, the divided data is updated. Finally, it is determined whether or not the final condition of learning is satisfied. Let us explain the algorithm of Table IV in detail.

As the initial condition, each of data $\boldsymbol{x}^l$ and $\boldsymbol{d}(\boldsymbol{x}^l)$, weights w_{ij} and v_{si} is divided into Q pieces and is sent to each server. On Step 3, each calculation of multiplying of weight and data is performed in each server and the result is sent to Server 0. On Step 4, each output y_i is calculated in Server 0, the output y_i is divided with Q pieces and sent to each server. On Step 5 of each server, multiplying $v_{si}^{(q)}$ and $y_i^{(q)}$ is performed and the result is sent to Server 0. On Step 6, each output $h_s(\boldsymbol{x}^l) (s \in Z_R)$ is calculated and divided randomly as $\sum_{q=1}^{Q} h_s^{(q)}(\boldsymbol{x}^l)$ in Server 0 and each of pieces is sent to each server. On Step 7, the error $d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)$ in each server is calculated and sent to Server 0. On Step 8, update amounts $p_{1(ij)}$ and $p_{2(si)}$ using the result of Step 7 are calculated in Server 0 and are sent to each server. On Step 9, each weight of $w_{ij}^{(q)}$ and $v_{si}^{(q)}$ is updated in each server. On Step 10, if t is a multiple of T_u, numbers $O_1^{(q)}$ and $O_0^{(q)}$ are selected randomly and sent to each server. On Step 11, if t is a multiple of T_u, each of $x_j^{l(q)}$ and $d_s^{(q)}(\boldsymbol{x}^l)$ is updated in each server and sent to Server 0. On Step 12 of Server 0, MSE is computed. On Step 13, if the final condition of learning is satisfied, the algorithm terminates otherwise go to Step 2 with $t \leftarrow t + 1$.

In the method proposed in Ref. [6], the divided data is fixed through learning. That is, in Table IV, the method is one that Steps 10 and 11 are removed. In the proposed method, by inserting Steps 10 and 11, the algorithm is generalized so that data division can be updated for each epoch in learning.

IV. NUMERICAL SIMULATIONS

A. Function Approximation

The simulation uses four systems specified by the following functions with $[0, 1]^4$ (for Eqs.(25) and (26)) and $[-1, 1]^4$ (for Eqs.(27) and (28)).

In this section, P is 10 and S is 1 in Eqs.(1) and (2). As simulation conditions, T is 50000, and constants K_w and K_v are 0.01. On the proposed method, assume that each divided data in each server is updated if the learning epoch is a multiple of 100, that is, T_u is 100. If the MSE of learning data is smaller than the threshold θ or the learning time is over the maximum number of learning times T, the algorithm terminates. Threshold θ is 0.0. As the initial condition, data and weights are divided into 3 pieces randomly, that is $Q =$

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE IV
GENERALIZED BP ALGORITHM FOR SDCD.

	Server 0	Server q ($q \in Z_Q$)
Initialize		Store $\{x_j^{l(q)} \mid l \in Z_L, j \in Z_n\}$ and $\{d_s^{(q)}(\boldsymbol{x}^l) \mid l \in Z_L, s \in Z_R\}$. Initialize $\{w_{ij}^{(q)} \mid i \in Z_P, j \in Z_n^*\}$ and $\{v_{si}^{(q)} \mid s \in Z_R, i \in Z_P^*\}$.
Step 1	$t \leftarrow 0$	
Step 2	Select a number $l \in Z_L$ randomly and send it to each server.	
Step 3		Calculate $w_{ij}^{(q)} x_j^{l(q)} (i \in Z_P, j \in Z_n^*)$and send them to Server 0.
Step 4	Calculate $y_i(\boldsymbol{x}^l)$ by using Eq.(10). Each y_i is divided as $\Pi_{q=1}^{Q} y_i^{(q)}$. Each data $y_i^{(q)} (q \in Z_Q)$ is sent to Server q.	
Step 5		Calculate $v_{si}^{(q)} y_i^{(q)} (s \in Z_R, i \in Z_P^*)$ and send them to Server 0.
Step 6	Calculate $h_s(\boldsymbol{x}^l) (s \in Z_S)$ by using Eq.(11). Each output $h_s(\boldsymbol{x}^l)$ is divided randomly as $\sum_{q=1}^{Q} h_s^{(q)}(\boldsymbol{x}^l)$. Each piece $h_s^{(q)}(\boldsymbol{x}^l) (q \in Z_Q)$ is sent to Server q.	
Step 7		Calculate $d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l) (s \in Z_R)$ and send them to Server 0.
Step 8	Calculate $p_{1(ij)} = \sum_{s=1}^{R} \sum_{q=1}^{Q} (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))(1 - h_s^{(q)}(\boldsymbol{x}^l)) v_{si} y_i(\boldsymbol{x}^l)(1 - y_i(\boldsymbol{x}^l)) \times (\Pi_{q=1}^{Q} w_{ij}^{(q)} x_j^{l(q)})$ and $p_{2(si)} = \sum_{q=1}^{Q} (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) h_s(\boldsymbol{x}^l)(1 - h_s(\boldsymbol{x}^l)) (\Pi_{q=1}^{Q} y_i^{(q)} v_{si}^{(q)})$ and send them to each server.	
Step 9		Each weight of $\{w_{ij}^{(q)} \mid i \in Z_P, j \in Z_j^*\}$ and $\{v_{si}^{(q)} \mid s \in Z_R, i \in Z_P^*\}$ is updated as $w_{ij}^{(q)} \leftarrow w_{ij}^{(q)} + \alpha p_{1(ij)} / w_{ij}^{(q)}$ $v_{si}^{(q)} \leftarrow v_{si}^{(q)} + \alpha p_{2(si)} / v_{si}^{(q)}$
Step 10	If t mod T_u is 0, $O_1^{(q)}$ and $O_0^{(q)}$ ($q \in Z_Q$) are selected randomly, where $1 = \Pi_{q=1}^{Q} O_1^{(q)}$ and $0 = \sum_{q=1}^{Q} O_0^{(q)}$ and sent them to each server.	
Step 11		If t mod T_u is 0, each weight of $\{x_j^{l(q)} \mid j \in Z_n, l \in Z_L\}$ and $\{d^{(q)}(\boldsymbol{x}^l) \mid l \in Z_L\}$ are update as follows : $x_j^{l(q)} \leftarrow O_1^{(q)} \times x_j^{l(q)}$ $d^{(q)}(\boldsymbol{x}^l) \leftarrow O_0^{(q)} + d^{(q)}(\boldsymbol{x}^l)$
Step 12	Calculate $E(X, W, V)$ by using Eq.(12).	
Step 13	If $E < \theta$ or $t < T$, algorithm terminates. Otherwise go to Step 2 with $t \leftarrow t + 1$.	

3. The numbers of learning and test data randomly selected are 1000 and 1000, respectively.

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{12} \quad (25)$$

$$y = \frac{\sin(2\pi x_1) \times \cos(x_2) \times \sin(\pi x_3) \times x_4 + 1.0}{2.0} \quad (26)$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{446.52} \quad (27)$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(3e^{3x_3} + 2e^{-4x_4})^{-0.5} - 0.077}{4.68} \quad (28)$$

After learning, Mean Square Error (MSE) for learning and test data for each method is compared. Table VII shows the results of simulations. In Table V, A means the conventional BP method, B means the BP method in Ref.[6] and C means the proposed method. MSE(Learning) and MSE(test) mean the MSE for learning and test, respectively. Each value in Table V means the MSE($\times 10^{-4}$) and it is average from twenty trials.

TABLE V
SIMULATION RESULTS FOR FUNCTION APPROXIMATION

		Eq(25)	Eq(26)	Eq(27)	Eq(28)
A	MSE(Learning)($\times 10^{-4}$)	0.45	9.76	0.57	0.83
	MSE(Test)($\times 10^{-4}$)	0.61	11.79	0.66	1.06
B	MSE(Learning)($\times 10^{-4}$)	0.57	15.72	0.74	1.41
	MSE(Test)($\times 10^{-4}$)	0.77	18.35	0.82	1.66
C	MSE(Learning)($\times 10^{-4}$)	0.60	13.61	0.87	1.54
	MSE(Test)($\times 10^{-4}$)	0.71	16.36	0.97	1.80

Table V shows the results of the comparison of accuracy for each method. In each box of Table V, Training and Test mean MSE of training and test ($\times 10^{-4}$), respectively. The result of the simulation is the average value from twenty trials. As shown in Table V, the proposed method shows almost the same accuracy compared to conventional and the proposed BP methods.

B. Pattern Classification

In this section, Iris, Wine, Sonar, BCW, and Spam data as shown in Table VI [11] are classified by using conventional and proposed methods. In Table VI, #data:L, #input : n,

and #output : R mean that the number of data, the number of input, and the number of output are L, n, and R, respectively. In this section, P is 10 and S is 3(for Iris and Wine) or 2(for Sonar, BCW, and Spam) in Eqs.(1) and (2). In the simulations, a 5-fold cross-validation method is used. As simulation condition, T is 50000, and constants K_w and K_v are 0.01. On the proposed method, assume that each divided data in each server is updated if the learning epoch is a multiple of 100, that is, T_u is 100. If the MSE of learning data is smaller than the threshold θ or the learning time is over the maximum number of learning times T, the algorithm terminates. Threshold θ is 3.0×10^{-2} for Iris and Wine, 4.0×10^{-2} for Sonar and BCW, and 8.0×10^{-2} for Spam, respectively. As the initial condition, data and weights are divided into 5 pieces randomly, that is $Q = 5$.

After learning, the rates of miss-classified data (RM) for learning and test data for each method are compared. Table VII shows the results of simulations. In Table VII, A means the conventional BP method, B means the BP method in Ref.[6] and C means the proposed method. RM(Learning) and RM(test) mean the rates of miss classified data for learning and test, respectively. Each value in Table VII mean the RM(%) and it is average from twenty trials.

TABLE VI
DATA USED IN SIMULATIONS

	Iris	Wine	Sonar	BCW	Spam
#data : L	150	178	208	683	4601
#input : n	4	13	60	9	57
#output : R	3	3	2	2	2

As shown in Table VII, the proposed method shows almost the same accuracy compared to conventional and the proposed BP methods.

As mentioned earlier, FL is a method of partitioning the dataset into multiple subsets and learning by distributed processing. This method is a method for preventing the concentration of learning data and preserving safety. However, in learning that performs distributed processing, it is necessary to repeatedly use the learning data itself, and the risk of data loss remains. On the other hand, in methods B and C introduced here, the learning data is used only when it is first divided and is not used for subsequent learning. In particular, in method B, this secretly divided data is fixed through learning. In method C, the divided data can be updated for each epoch in learning. From these facts, it is considered that the proposed methods B and C are safer than the conventional FL, and in particular, method C is safer than B.

V. CONCLUSION

In this paper, we proposed the generalized BP learning method for edge computing. The feature of the proposed method is that learning is performed using the divided data without using the learning data itself. In the conventional FL, data is partitioned into subsets, and learning is performed by distributed processing. In this case, the learning data itself is used in the calculation of each server. In the proposed method, after the learning data is divided once, the distributed learning using the divided data is performed. In particular, we proposed a method of repeating data division multiple times through learning, which is a generalized learning method. In numerical simulations, the proposed method shows almost the same accuracy compared to conventional methods. In future works, we will propose other learning methods with divided data for edge computing.

TABLE VII
SIMULATION RESULTS FOR PATTERN CLASSIFICATION

		Iris	Wine	Sonar	BCW	Spam
A	RM(Learning)(%)	2.08	0.52	0.91	2.36	4.60
	RM(Test)(%)	2.93	1.78	17.83	3.01	6.43
B	RM(Learning)(%)	1.90	1.06	2.66	2.26	4.66
	RM(Test)(%)	2.12	2.51	17.80	2.48	4.81
C	RM(Learning)(%)	1.80	1.18	2.81	2.20	4.72
	RM(Test)(%)	4.43	3.81	19.26	3.14	5.86

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things : A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communication Surveys & Tutorials*, vol.17, no.4, pp.2347-2376, 2015.

[2] J. Chen, and X. Ran, "Deep Learning with Edge Computing: A Review," *Proc. of the IEEE*, vol. 107, no. 8, 2019.

[3] M. G. Sarwar Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayana, F. and Hussain, "Machine Learning at the Network Edge: A Survey," *arXiv:1908.00080 [cs.LG]*, 2019.

[4] Q. Yang, Y. Li, T. Chen, and Y. Tong, "Federated Machine Learaning : Concept and Applications," *ACM Trans. Intell. Syst. Technol.*, vol.10, no.2, Article 12, 2019.

[5] 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*, pp.450-454, 2019

[6] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, 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.

[7] D. Evans, V. Kolesnikov, and M. Rosulek, "A Pragmatic Introduction to Secure Multi-Party Computation," *Foundations and Trends in Privacy and Security*, vol. 2, issue 2-3, pp.70-246, 2018.

[8] C. C. Aggarwal, and P. S. Yu, "Privacy Preserving Data Mining: Models and Algorithms," ISBN 978-0-387-70991-8, Springer-Verlag, 2009.

[9] Y. Miyanishi, A. Kanaoka, F. Sato, X. Han, S. Kitagami, Y. Urano, and N. Shiratori, "New Methods to Ensure Security to Increase User's Sense of Safety in Cloud Services," *Proc. of The 14th IEEE Int. Conference on Scalable Computing and Communications (ScalCom-2014)*, pp.859-865, 2014.

[10] M. M. Gupta, L. Jin, and N. Honma, "Static and Dynamic Neural Networks", *IEEE Pres, Wiley-Interscience*, 2003.

[11] UCI Repository of Machine Learning Databases and Domain Theories, https://archive.ics.uci.edu/ml/datasets.php.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)