Securely Distributed Computation with Divided Data for Particle Swarm Optimization

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

***Abstract*—Machine Learning (ML) on cloud and edge systems is widely used to analyze big data and deal with complex problems. On the other hand, inadequate data security management raises many privacy issues. Therefore, there is a lot of interest in studying secure machine learning on cloud and edge systems. For one of these, many studies have been conducted to distribute learning data into multiple servers and realize machine learning by distributed processing. Federated Learning (FL) is one of them. The authors have proposed a method in which learning data are divided into multiple pieces in advance, multiple pieces are distributed to each server, and learning is realized by distributed processing. In the previous papers, we proposed the methods that used local search based on Steepest Descent Method (SDM) such as Back Propagation (BP) and K-means. In this paper, we will propose a securely distributed computation with divided data as a learning method that has both global and local search like Particle Swarm Optimization (PSO).**

***Index Terms*—Cloud and edge systems, Machine learning, Particle Swarm Optimization, Secure Distributed Computation with Divided data.**

I. Introduction

With advances in artificial intelligence (AI), many studies have been done with machine learning (ML). In particular, ML on cloud and edge systems is widely used to analyze big data and deal with complex problems. In addition, the spreading of cloud and edge systems allows uses such as big data analysis to analyze enormous information accumulated by users [1], [2]. On the other hand, users of cloud and edge computing cannot escape from anxiety about the possibility of information being abused or leaked. How is the computing system constructed to avoid the above risk? Data encryption is one of the typical approaches [3]. It is an effective method, but the encrypted data must be decrypted to get plain texts when processed in cloud and edge systems. Therefore, a safe system for distributed processing with partitioned or divided data attracts attention, and a lot of studies have been done [4], [5]. One of them is the securely distributed computation with divided learning data (SDCD). BP and NG methods have been proposed as machine learning methods using it [5], [6]. On the other hand, these methods are ones using local search for finding solutions. It is desirable to combine global and local searches to improve the accuracy of the solution. Swarm intelligence is known as a method for easily realizing a global search, and by incorporating this mechanism into the solution search, a highly accurate search of the solution is expected [7], [8]. However, little study has been done on a secret calculation that uses a global and local search of solutions such as swarm intelligence.

In this paper, we propose a learning algorithm using securely distributed processing with divided data for Particle Swarm Optimization (PSO), which is a learning method for swarm intelligence. In particular, a learning method for Hybrid Particle Swarm Optimization (HPSO) is proposed, where HPSO is a combination of BP and PSO. Its effectiveness is shown 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 configuration for cloud and edge systems

Let us explain the outline of distributed processing. Figure 1 shows an example of the system. The system is composed of L terminals and $Q+1$ servers. Let x and f be a (scalar) data and a function, respectively. Each data x is divided into some pieces and each piece is sent to each server. In the case of Figure 1, each data is divided into the addition form.

First, each data x from 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 the result is 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 cannot be obtained in one process, multiple processes are repeated.

The problem is how to determine function f_q in each server.

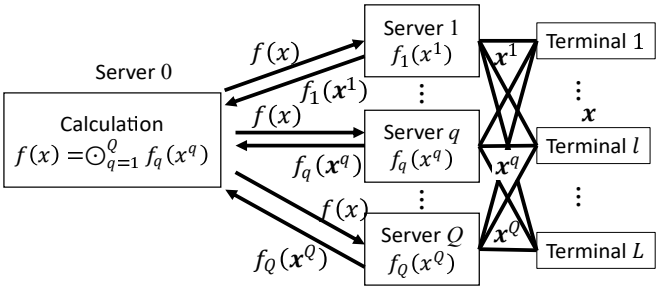

Fig. 1. An example of the proposed system : The data given from each terminal is divided into Q pieces and each piece is sent to each server. At server $q \in Z_Q$, $f_q(x^q)$ is calculated and sent to Server 0. On Server 0, the result $f(x)$ is calculated by using $f_1(x^1),\cdots,f_Q(x_Q)$.

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

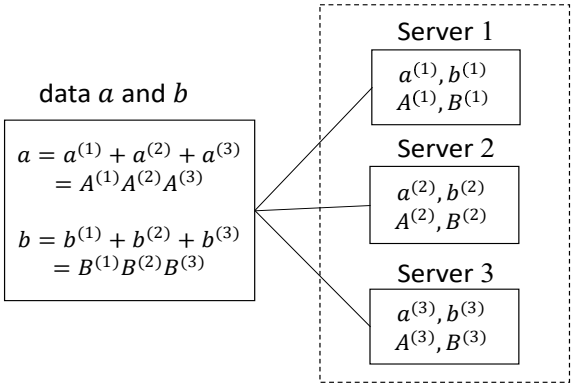

Fig. 2. The representation of secure divided data.

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

	data a	data b	addition
Server 1	$a^{(1)}$ =4	$b^{(1)}$ =3	7
Server 2	$a^{(2)}$ =-1	$b^{(2)}$ =2	1
Server 3	$a^{(3)}$ =3	$b^{(3)}$ =3	6
addition	6	8	14

B. Data division for the proposed method

Figure 2 shows the representation of divided data, which is used in the proposed method [6]. In Figure 2, 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 randomly, respectively.

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, where $a^{(q)}$, $b^{(q)}$, $A^{(q)}$ and $B^{(q)}$ for $q \in \{1, 2, 3\}$ are the random numbers.

By using them, four basic operations of arithmetic (addition, subtraction, multiplication, and division) can be calculated without decrypting a and b as follows :
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)$a/b = (A^{(1)}/B^{(1)})(A^{(2)}/B^{(2)})(A^{(3)}/B^{(3)})$

In this case, any server cannot know a and b themselves.

For example, let $a = 6$ and $b = 8$. a and b are divided as $6 = 4 + (-1) + 3 = (-1)\times 2\times(-3)$ and $8 = 3 + 2 + 3 = 2\times(-4)\times(-1)$. In this case, $a^{(1)} = 4$, $a^{(2)} = -1$, $a^{(3)} = 3$, $A^{(1)} = -1$, $A^{(2)} = 2$, $A^{(3)} = -3$, $b^{(1)} = 3$, $b^{(2)} = 2$, $b^{(3)} = 3$, $B^{(1)} = 2$, $B^{(2)} = -4$ and $B^{(3)} = -1$, respectively. $a + b$ can be calculated as follows (See Table I).

$$\begin{aligned} a + b &= (a^{(1)} + b^{(1)}) + (a^{(2)} + b^{(2)}) + (a^{(3)} + b^{(3)}) \\ &= (4 + 3) + ((-1) + 2) + (3 + 3) = 14 \end{aligned}$$

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

$$\begin{aligned} a\times b &= (A^{(1)}\times B^{(1)})(A^{(2)}\times B^{(2)})(A^{(3)}\times B^{(3)}) \\ &= ((-1)\times 2)(2\times(-4))((-3)\times(-1)) = 48 \end{aligned}$$

C. Neural Network and BP method

In this section, let us explain the conventional three-layered Neural Network (NN) as shown in Fig.3 and BP method [9]. For any positive integer i, let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$. The function $\boldsymbol{h} : J_{in}^n \to J_{out}^R$ is determined for each input $\boldsymbol{x} \in J_{in}^n$ 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, The set of learning data, $X = \{(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l)) | \boldsymbol{x}^l \in J_{in}^n, \boldsymbol{d}(\boldsymbol{x}^l) \in J_{out}^R, l \in Z_L\}$, is used to determine the weights, 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 $\boldsymbol{x}^l$.

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	$A^{(2)}$ =2	$B^{(2)}$ =-4	-8
Server 3	$A^{(3)}$ =-3	$B^{(3)}$ =-1	3
multiplication	6	8	48

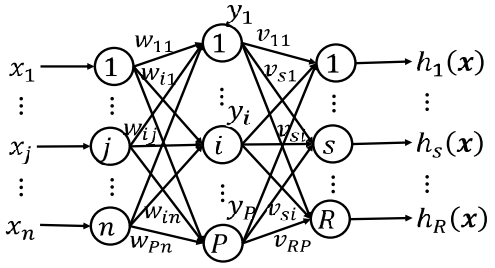

Fig. 3. An example of three layered Neural Network

Let $W = \{w_{ij} | i \in Z_P, j \in Z_n^*\}$ and $V = \{v_{si} | s \in Z_R, i \in Z_P^*\}$ be the sets of weights. 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 $y_0 = 1$.

To determine the weights, the Mean Square Error (MSE) for the learning data is used as the evaluation function for the BP method. In this case, the evaluation function is defined as follow:

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

where X, W, and V are the sets of learning data and weights, respectively.

The purpose of the BP method is to minimize the evaluation function of Eq.(3). By using the BP method, each weight of W and V is determined as follows [9]. In the following, X, T, θ, and α mean the set of leaning data, the maximum number of learning time, threshold, and learning rate, respectively.

[**B**P method] BP(X, W, V, T)
Input : the set $X = \{(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l)) | l \in Z_L\}$ of learning data
Output : the sets $W = \{w_{ij} | i \in Z_P, j \in Z_n^*\}$ 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, respectively.

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

[**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 $t\leftarrow t+1$.

□

D. PSO to determine weights of NN

Particle Swarm Optimization (PSO) is a method to solve the optimization problem. Let us explain it. It finds the value of the determinant $\boldsymbol{x}$ that maximizes the nonlinear objective function $g(\boldsymbol{x})$. To solve this problem, PSO prepares multiple individuals $P = \{P_k | k\in Z_m\}$ and performs a multipoint search that each solution is found in each individual. Let m be the number of individuals, assign a decision variable (solution candidate) to each individual P_k, and let this be $\boldsymbol{x}_k$. The solution search method updates the solution candidates by using the self-best solution $\boldsymbol{p}_k$ found by each individual P_k in the past search and the past best solution $\boldsymbol{a}$ found in the whole. By approaching the solution $\boldsymbol{x}_k$ in each individual closer to two excellent solutions $\boldsymbol{a}$ and $\boldsymbol{p}_k$, an excellent solution with a large objective function value can be expected.

The following is the algorithm of PSO for learning of NN [7]. The purpose of the algorithm is to determine the weights W and V for NN so that the MSE E for the learning data X is minimized. In the following, the maximum number of learning times T, threshold θ, and constant values c_0, c_1, and c_2 are determined in advance, respectively. For $k\in Z_m$, a set $\{W^k, V^k\}$ corresponds to $\boldsymbol{x}_k$ in the above explanation of PSO. The set, $\{W^{k*}, V^{k*}\}$ corresponds to self-best solution $\boldsymbol{p}_k$. The set $\{W^K, V^K\}$ corresponds to the best solution $\boldsymbol{a}$ (See Table III). The function $E(X, W^k, V^k)$ is the MSE for learning data X when the weights W^k and V^k are used.

[**A**lgorithm of PSO] [8]

Input : the set X of learning data

Output : the sets W and V of weights

[**S**tep 1]

Initialize each weight of W and V. Set $t\leftarrow 0$, $W^{k^*}\leftarrow W^k$, $V^{k^*}\leftarrow V^k$ and $K\leftarrow \arg\ \min_{k^*}\{E(X, W^{k^*}, V^{k^*})\}$.

[**S**tep 2]

Select the numbers r_1 and r_2 randomly.

[**S**tep 3]

For $k\in Z_m$, update $w_{ij}^k\in W^k$ and $v_i^k\in V^k$ as follows:

$$\begin{aligned} \triangle w_{ij}^k &\leftarrow c_0\triangle w_{ij}^k + c_1 r_1(w_{ij}^{k^*} - w_{ij}^k) + c_2 r_2(w_{ij}^K - w_{ij}^k) \\ w_{ij}^k &\leftarrow w_{ij}^k + \triangle w_{ij}^k \qquad (6) \\ \triangle v_{si}^k &\leftarrow c_0\triangle v_{si}^k + c_1 r_1(v_{si}^{k^*} - v_{si}^k) + c_2 r_2(v_{si}^K - v_{si}^k) \\ v_{si}^k &\leftarrow v_{si}^k + \triangle v_{si}^k \qquad (7) \end{aligned}$$

[**S**tep 4]

For $k\in Z_m$, if $E(X, W^k, V^k) < E(X, W^{k^*}, V^{k^*})$, then set $W^{k^*}\leftarrow W^k$ and $V^{k^*}\leftarrow V^k$.

[**S**tep 5]

Set $K\leftarrow \arg\ \min_{k^*}\{E(X, W^{k^*}, V^{k^*})\}$, $W^*\leftarrow W^{K^*}$ and $V^*\leftarrow V^{K^*}$.

[**S**tep 6]

If $E(X, W^*, V^*) < \theta$ or $t > T$, then the algorithm terminates. Otherwise, go to Step 2 with $t\leftarrow t+1$.

□

In Step 3, each weight of $\{W^k, V^k\}$ is updated by using the current weights w_{ij}^k and v_{si}^k, the self-best weights w_{ij}^{k*} and v_{si}^{k*} and the past best weights w_{ij}^K and v_{si}^K in the whole. In Step 4, the self-best weight $\{W^{k^*}, V^{k^*}\}$ in the past search for the solution $\{W^k, V^k\}$ is updated. In Step 5, the past best (optimal) solution $\{W^K, V^K\}$ in the whole is updated.

On the other hand, PSO needs much time compared to the Steepest Descent method (SDM) like the BP method. To improve it, Hybrid Particle Swarm Optimization (HPSO) is proposed [8]. As HPSO uses both PSO and BP, the optimal solution is calculated efficiently and needs a short time compared to the usual PSO. The algorithm HPSO is obtained by replacing Step 3 of the algorithm PSO with the following Step 3'.

[**S**tep 3']

For $k\in Z_m$

Update W^k, V^k, W^{k*} and V^{k*} by using Eqs.(6) and (7).

$BP^*(X, W^k, V^k, T_{BP})$

$BP^*(X, W^{k*}, V^{k*}, T_{BP})$

In Step 3', $BP^*(X, W^k, V^k, T_{BP})$ and $BP^*(X, W^{k*}, V^{k*}, T_{BP})$ are the method $BP(X, W, V, T)$ with $W = W^k$, $V = V^k$ and $T = T_{BP}$, and $W = W^{k*}$, $V = V^{k*}$ and $T = T_{BP}$, respectively.

III. Securely distributed system with divided data for HPSO

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:

$$x_j^l = \Pi_{q=1}^Q x_j^{l(q)} \qquad (8)$$

$$d_s(\boldsymbol{x}^l) = \sum_{q=1}^{Q} d_s^{(q)}(\boldsymbol{x}^l) \qquad (9)$$

On the same way, each weight of the sets W and V is divided with Q pieces and they are stored in each server as follows :

$$w_{ij} = \Pi_{q=1}^Q w_{ij}^{(q)} \qquad (10)$$

$$v_{si} = \Pi_{q=1}^Q v_{si}^{(q)} \qquad (11)$$

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

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

TABLE III
RELATION AMONG WEIGHTS FOR PSO

Individual	Solution candidate $\boldsymbol{x}_k$	Self-best $\boldsymbol{p}_k$	Best in whole $\boldsymbol{a}$
P_1	$\{W^1, V^1\}$	$\{W^{1*}, V^{1*}\}$	
$\vdots$			
P_k	$\{W^k, V^k\}$	$\{W^{k*}, V^{k*}\}$	$\{W^K, V^K\}$
$\vdots$			
P_m	$\{W^m, V^m\}$	$\{W^{m*}, V^{m*}\}$	

Let $\Pi_{q=1}^{Q} x_0^{(q)} = 1$. An output of the NN for the divided data based on Eqs.(8), (9), (10) and (11) is calculated instead of (1) and (2) as follows :

$$y_i(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{j=0}^{n}\Pi_{q=1}^{Q} w_{ij}^{(q)} x_j^{(q)}\right)\right))} \tag{12}$$

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

$$h_s(\boldsymbol{x}) = \frac{1}{1+\exp\left(-\left(\sum_{i=0}^{P}\Pi_{q=1}^{Q} v_{si}^{(q)} y_i^{(q)}\right)\right)} \tag{13}$$

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 :

$$E(X) = \frac{1}{2L}\sum_{l=1}^{L}\sum_{s=1}^{R}\left(\sum_{q=1}^{Q}(d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))\right)^2 \tag{14}$$

Based on BP, 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 [5] :

$$\begin{aligned} w_{ij}^{(q)} = w_{ij}^{(q)} + \quad & \alpha \quad \sum_{s=1}^{R}\sum_{q=1}^{Q}(d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))(1 - h_s^{(q)}(\boldsymbol{x}^l)) \\ & \times \quad \Pi_{q=1}^{Q} v_{si}^{(q)} h_s^{(q)}(\boldsymbol{x}^l)(1 - h_s(\boldsymbol{x}^l)) \\ & \times (\Pi_{q=1}^{Q} w_{ij}^{(q)} x_j^{l(q)})/w_{ij}^{(q)} \end{aligned} \tag{15}$$

$$\begin{aligned} v_{si}^{(q)} = v_{si}^{(q)} + \alpha & \sum_{q=1}^{Q}(d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) h_s(\boldsymbol{x}^l) \\ & \times (1 - h_s(\boldsymbol{x}^l))(\Pi_{q=1}^{Q} y_i^{(q)} v_{si}^{(q)})/v_{si}^{(q)}) \end{aligned} \tag{16}$$

Tables III and IV show the relation among weights of HPSO and divided data. Table III shows how to divide weights for each NN solution, self-best solution, and overall best solution of PSO and store them in the server. For example, it shows that any element w^k of W^k is divided into $w^{k(1)} + \cdots + w^{k(Q)}$ and stored in Q servers, respectively.

Table V shows the algorithm of BP. In Table V, each component of divided data and parameters (weights) is stored in each server. In Step 3, the product of the l-th data and the weight for the first layer is calculated and sent to Server 0. In Steps 4 , 5, and 6, the product of the input and the weight in the second layer is calculated and the result is divided into Q pieces and sent to each server. In Steps 7 and 8,

TABLE IV
RELATION AMONG WEIGHTS FOR DIVIDED SOLUTION CANDIDATE WITH EACH SERVER

Server	The set of partitions of divided solution candidate
Server 1	$\{W^{k(1)}, V^{k(1)} \mid k \in Z_m\}$
	$\{W^{k*(1)}, V^{k*(1)} \mid k \in Z_m\}$
	$\{W^{K(1)}, V^{K(1)}\}$
$\vdots$	
Server q	$\{W^{k(q)}, V^{k(q)} \mid k \in Z_m\}$
	$\{W^{k*(q)}, V^{k*(q)} \mid k \in Z_m\}$
	$\{W^{K(q)}, V^{K(q)}\}$
$\vdots$	
Server Q	$\{W^{k(Q)}, V^{k(Q)} \mid k \in Z_m\}$
	$\{W^{k*(Q)}, V^{k*(Q)} \mid k \in Z_m\}$
	$\{W^{K(Q)}, V^{K(Q)}\}$

the difference between the desired and the output results are calculated in each server and they are integrated at Server 0 to calculate the updated amount of the weight. In Step 9, the weight in each server is used to update the weight of each server. In Steps 11 and 12, the algorithm termination condition is checked.

B. HPSO for SDCD

In the case of PSO, each piece of $w_{ij}^{k(q)}$ and $v_{si}^{k(q)}$ of divided data for weights $w_{ij}^{k} (i \in Z_P, j \in Z_P^*)$ and $v_{si}^{k} (s \in Z_R, i \in Z_P^*)$ is updated by using Eqs.(17) and (18) instead of Eqs.(6) and (7).

$$\begin{aligned} \triangle w_{ij}^{k(q)} &= c_0 \triangle w_{ij}^{k(q)} + c_1 r_1 (w_{ij}^{k^*(q)} - w_{ij}^{k(q)}) \\ &\quad + c_2 r_2 (w_{ij}^{K(q)} - w_{ij}^{k(q)}) \\ w_{ij}^{k(q)} &= w_{ij}^{k(q)} + \triangle w_{ij}^{k(q)} (j \in Z_n^*, i \in Z_P) \end{aligned} \tag{17}$$

$$\begin{aligned} \triangle v_{si}^{k(q)} &= c_0 \triangle v_{si}^{k(q)} + c_1 r_1 (v_{si}^{k^*(q)} - v_{si}^{k(q)}) \\ &\quad + c_2 r_2 (v_{si}^{K(q)} - v_{si}^{k(q)}) \\ v_{si}^{k(q)} &= v_{si}^{k(q)} + \triangle v_{si}^{k(q)} (s \in Z_R, i \in Z_P^*) \end{aligned} \tag{18}$$

Let $W^{k(q)} = \{w_{ij}^{k(q)} \mid i \in Z_P, j \in Z_n^*\}$ and $V^{k(q)} = \{v_{si}^{k(q)} \mid s \in Z_S, i \in Z_P^*\}$ for the sets $\{W^k, V^k\}$. The general flow of the proposed method is shown as follows :

[**T**he general flow of HPSO] [8]
Input : the set $X = \{x_j^{l(q)}, d_s^{(q)}(\boldsymbol{x}^l) \mid l \in Z_L, j \in Z_n, q \in Z_Q\}$
Output : the set of best in whole $\{W^{K(q)}, V^{K(q)} \mid q \in Z_Q\}$
T:Maximum number of learning epoch for PSO
T_{BP}:Maximum number of learning epoch for BP
1 : Initialize the sets $\{W^{k(q)}, V^{k(q)} \mid k \in Z_m, q \in Z_Q\}$
2 : $t \leftarrow 0$
3 : while $t < T$ do
  3.1 : Calculate the self-best $\{W^{k*(q)}, V^{k*(q)}\}$ for $\{W^{k(q)}, V^{k(q)}\}$ and global-best $\{W^{K*(q)}, V^{K*(q)}\}$.
  3.2 : Update $\{W^{k(q)}, V^{k(q)}\}$ by using Eqs.(17) and (18)
  3.3 : For $k \in Z_m$
    3.3.1 : $t_{BP} \leftarrow 0$
    3.3.2 : while $t_{BP} < T_{BP}$ do
      3.3.2.1 : Update $\{W^{k(q)}, V^{k(q)}\}$ by using Eqs.(15) and (16)
      3.3.2.2 : $t_{BP} \leftarrow t_{BP} + 1$
    End

TABLE V
BP METHOD FOR SDCD : $BP(X, W^{(q)}, V^{(q)}, T)$

	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.(12). 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.(13). Divide $h_s(\boldsymbol{x}^l) = \sum_{q=1}^{Q} h_s^{(q)}(\boldsymbol{x}^l)$. 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.(14).	
Step 11	If $E < \theta$ or $t < T$, algorithm terminates. Otherwise go to Step 2 with $t \leftarrow t + 1$.	

End
3.3.3 : $t \leftarrow t + 1$
End

□

The proposed HPSO is shown in Table VI. In Table VI, $E(X, \{W^{k(q)}, V^{k(q)}\})$ means the calculation result of Eq.(14) when $\{W^{k(q)}, V^{k(q)}\}$ is used as $\{W^{(q)}, V^{(q)}\}$.

Each server has arbitrary components of W^k and V^k that are divided Q pieces in advance. In Steps 1 and 2, the self-best solution candidates and the overall best solution are set. In Steps 3 and 4, $w_{ij}^{k(q)}$ and $v_{si}^{k(q)}$ are updated based on the update formula of PSO. In Step 5, each weight for m NNs is updated based on the BP method in Table III. In Steps 6 and 7, self-best $\{W^{k*(q)}, V^{k*(q)}\}$ is updated. In Step 8, the overall best $\{W^{K(q)}, V^{K(q)}\}$ is updated. In Step 9, the final condition is checked.

IV. NUMERICAL SIMULATIONS FOR THE PROPOSED ALGORITHM

In this section, numerical simulations for function approximation and pattern classification are performed. The BP, HPSO, and BP for SDCD mean the conventional BP and HPSO methods and the conventional BP for SDCD, respectively. The Proposed means the proposed method.

A. Function Approximation

This simulation uses four systems specified by the following functions with $[0, 1]^2$.

$$y = \sin(\pi x_1^3) x_2 \tag{19}$$

$$y = \frac{\sin(2\pi x_1^3)\cos(\pi x_2) + 1}{2} \tag{20}$$

$$y = \frac{1.9(1.35 + \exp(x_1)\sin(13(x_1 - 0.6)^2)}{7} \times \exp(-x_2)\sin(7x_2)) \tag{21}$$

$$y = \frac{\sin(10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2) + 1}{2} \tag{22}$$

The numbers of data for learning and test are 100, respectively. Each input data is selected randomly. The simulation conditions are $P = 10$ for NN, $T = 100000$ for BP, $T = 10$, $T_l = 10000$ and $m = 10$ for HPSO and the proposed method, $Q = 3$ for the proposed method, respectively. The threshold is $\theta = 0.0$ for all methods.

Table VII shows the comparison of accuracy for each method. In each box of Table VII, Learn and Test mean MSE of learning and test, respectively. Each result of the simulations is the average value from twenty trials. In Table VII, the proposed method shows high accuracy compared to BP methods and almost the same accuracy compared to the conventional HPSO.

B. Pattern Classification

Let us show the result for pattern classification using benchmark problems of Iris, Wine, Sonar, and BCW in the UCI database[10] as shown in Table VIII. In Table VIII, #data, #input, and #class mean the numbers of data, input, and class for each data, respectively. In this simulation, 5-fold cross-validation as the evaluation method is used. The simulation conditions are $P = 10$ for NN, $T = 10000$ for BP and BP for SDCD as shown in Table V, $T = 10$, $T_l = 10000$ and $m = 10$ for HPSO and the proposed method, $Q = 3$ for

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

TABLE VI
PROPOSED HPSO FOR SDCD

	Server 0	Server$q$ $(1\leq q\leq Q)$
Initialize		Initialize $\{W^{k(q)}, V^{k(q)} \mid k\in Z_m\}$
Step 1		$W^{k^*(q)}\leftarrow W^{k(q)}$, $V^{k^*(q)}\leftarrow V^{k(q)} (k\in Z_m)$
Step 2	Calculate $K\leftarrow \arg\min_{k^*}\{E(X,\{W^{k^*(q)}, V^{k^*(q)})\}\}$ and send it to each server. $t\leftarrow 0$.	
Step 3	Select two numbers r_1 and r_2 randomly and send them to each server.	
Step 4		By using Eqs.(17) and (18), each of $\{w_{ij}^{k(q)} \mid i\in Z_P, j\in Z_j^*\}$ and $\{v_i^{k(q)} \mid i\in Z_P^*\}$ is updated$(k\in Z_m)$.
Step 5	$BP(X, W^{k(q)}, V^{k(q)}, T_{BP})$ and $BP(X, W^{k*(q)}, V^{k*(q)}, T_{BP})$ for $k\in Z_m$: Update $\{W^{k(q)}, V^{k(q)}\}$ and $\{W^{k*(q)}, V^{k*(q)}\}$ for $k\in Z_m$ by using BP of Table V.	
Step 6	By using Eq.(14), $E(X,\{W^{k(q)}, V^{k(q)}\})$ and $E(X,\{W^{k^*(q)}, V^{k^*(q)})\}$ are calculated. If $E(X,\{W^{k(q)}, V^{k(q)}\}) < E(X,\{W^{k^*(q)}, V^{k^*(q)}\})$, then go to Step 7. Otherwise go to Step 8.	
Step 7		Set $W^{k^*(q)}\leftarrow W^{k(q)}$, $V^{k^*(q)}\leftarrow V^{k(q)} (k\in Z_m)$.
Step 8	Calculate $K\leftarrow \arg\min_{k^*}\{E(X, W^{k^*}, V^{k^*})\}$ and send it to each server.	
Step 9	If $E(X, W^{K(q)}, V^{K(q)}) < \theta$ or $t > T$, then the algorithm terminates. Otherwise, go to Step 3 with $t\leftarrow t+1$.	

TABLE VII
RESULT FOR FUNCTION APPROXIMATION $(\times 10^{-4})$

		Eq.(19)	Eq.(20)	Eq.(21)	Eq.(22)
BP	Learn	1.40	4.22	10.82	39.55
	Test	2.33	13.33	23.05	88.75
HPSO	Learn	0.99	2.93	4.07	9.10
	Test	2.88	10.79	13.31	40.79
BP for SDCD	Learn	3.39	9.59	21.26	64.32
	Test	10.57	16.92	48.24	186.00
Proposed for HPSO	Learn	0.64	2.84	4.34	9.72
	Test	6.57	11.32	12.79	75.96

TABLE VIII
THE DATASET FOR PATTERN CLASSIFICATION

	Iris	Wine	Sonar	BCW
# data	150	178	208	683
# input	4	13	60	9
# class	3	3	2	2

the proposed method, respectively. The threshold is $\theta = 0.03$ for Iris and Wine and $\theta = 0.04$ for Sonar and BCW.

Table IX shows the result of the comparison between the conventional and the proposed methods. In each box of Table IX, Learn and Test mean the rate (%) of misclassified data for learning and test, respectively. Each value is average from twenty trials. In Table IX, the proposed method shows almost the same accuracy compared to others.

V. CONCLUSION

In this paper, we proposed the HPSO learning method for securely distributed processing with divided data. The feature of the proposed method is a learning method using global and local search and learning is performed using the divided data without using the learning data itself. In the proposed method, after each learning data is divided and stored in each server once, HPSO is performed by using the divided data without using the original data. In numerical simulations, the proposed method shows almost the same accuracy compared to conventional HPSO. In future works, we will propose other learning methods with a global search using securely distributed processing with divided data.

TABLE IX
RESULT FOR PATTERN CLASSIFICATION (%)

		Iris	Wine	Sonar	BCW
BP	Learn	3.58	1.91	1.22	2.33
	Test	3.83	5.06	16.88	2.91
HPSO	Learn	3.44	0.71	0.23	1.60
	Test	3.60	4.08	15.93	3.58
BP for SDCD	Learn	3.59	3.39	1.97	2.26
	Test	3.87	5.61	18.62	2.95
Proposed for HPSO	Learn	2.62	0.37	0.31	0.96
	Test	4.47	4.89	19.17	4.01

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, X. Ran, "Deep Learning with Edge Computing : A Review," *Proc. of the IEEE*, vol.107, no.8, 2019.
[3] M. Tebaa, S. E. Hajji, A. E. Fhazi, "Homomorphic Encryption Applied to the Cloud Security," *Proc. of the World Congress on Engineering 2012*, vol.1, ISSN:2078-0966, 2012.
[4] Q. Yang, Y. Liu, T. Chen and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, 2019.
[5] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, 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] Y. Miyanishi, A. Kanaoka, F. Sato, X. Han, S. Kitagami, Y. Urano, 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.
[7] J. Kennedy and R. C. Eberhart, "Partucle swarm optimization," *Proc. IEEE Int. Conf. Neural Networks*, pp.1942-1948, 1995.
[8] S. Kathpal, R. Vohra, J. Singh, R. S. Sawhey, "Hybrid PSO-SA Algorithm for Achieving Partitioning Optimization in Various Network Applications," *Procedia Engineering*, vol.38, pp.1728-1734, 2012.
[9] M. M. Gupta, L. Jin, N. Honma, "Static and Dynamic Neural Networks," *IEEE Pres, Wiley-Interscience*, 2003.
[10] UCI Repository of Machine Learning Databases and Domain Theories, https://archive.ics.uci.edu/ml/datasets.php.