Securely Distributed Computation with Divided Data and Parameters for Hybrid 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 address complex problems. On the other hand, many privacy issues have arisen due to inadequate data security controls. Therefore, studies on secure machine learning on cloud and edge systems have attracted much attention. One of such studies is Federated Learning (FL), in which data is distributed on multiple servers to achieve machine learning through distributed processing. We have proposed a method to realize learning by distributed processing on multiple servers by dividing the learning data and parameters into multiple pieces in advance and distributing these pieces to each server. In previous papers, we have proposed a learning method using local search based on the Steepest Descent Method (SDM) such as Back Propagation (BP) and Neural Gas (NG). In this paper, we propose a learning method that combines a global and local search for solutions for secure distributed processing computation. In particular, we propose a secret distributed processing method for Hybrid Particle Swarm Optimization (HPSO) which is one of the global and local search methods. Its effectiveness is demonstrated by numerical simulations.**

***Index Terms*—Cloud and Edge Systems, Machine Learning, Particle Swarm Optimization, Secure Distributed Computation, Divided Data and Parameters, Back Propagation, Neural Gas.**

I. Introduction

With advances in artificial intelligence (AI), many studies have been conducted using machine learning (ML). In particular, ML on cloud and edge systems has been widely used to analyze big data and deal with complex problems. The widespread use of cloud and edge systems has also enabled the use of big data analysis and other applications for analyzing huge amounts of information accumulated by users [1], [2]. On the other hand, users of cloud and edge systems cannot escape the fear that their information may be misused or leaked. How can secure computation be constructed to avoid such risks? Data encryption is one typical approach [3], [6]. While it is an effective means, when processing in the cloud and edge systems, the encrypted data must be decrypted to obtain the plaintext. As an alternative, secure distributed processing computation using subsets or divided data has attracted much attention and has been the subject of many studies [4], [5], [8]. One such method is secure distributed computation with divided learning data (SDCD). BP and NG methods have been proposed as ML methods that preserve data confidentiality using this method [7], [8]. On the other hand, although these methods are fast, the accuracy of the solution is not promising because they only use local search. Hence, it is desirable to develop a distributed processing computation method that preserves the confidentiality of methods that combine global and local searches for solutions. Hybrid Particle Swarm Optimization (HPSO) is known as an effective solution search method that combines PSO, a global search method, and BP or NG, a local search method.

In this paper, we propose a secure distributed learning method for HPSO with divided data. Its effectiveness is demonstrated by numerical simulations of function approximation and pattern classification.

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

Noritaka Shigei is a Professor of Kagoshima University, 1-21-24, Korimoto, Kagoshima, Japan (e-mail: shigei@ibe.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

We present the concept of distributed processing. In Fig. 1, we show 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 multiple pieces and each piece is sent to a server. In the case of Fig. 1, each data is divided in the addition form.

First, each data x from each terminal is divided into Q pieces randomly as $x = \sum_{q=1}^{Q} x^q$. The q-th piece x^q is sent to Server q. The 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, $f_q(x^q)$ is 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.

B. Data division for the proposed method

In Fig. 2, we show the representation of divided data, which is used in the proposed method [8]. In Fig. 2, let a and b be two positive integers and the number of servers is 3. We first divide integers a and b into three pieces of real numbers randomly, respectively. For any positive integer i, let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$.

Let $a = \sum_{q=1}^{Q} a^{(q)}$ and $b = \sum_{q=1}^{Q} b^{(q)}$ as the addition form and $a = \Pi_{q=1}^{Q} A^{(q)}$ and $b = \Pi_{q=1}^{Q} B^{(q)}$ as the multiplication form, where $a^{(q)}$, $b^{(q)}$, $A^{(q)}$ and $B^{(q)}$ for $q \in Z_Q$ are the random numbers.

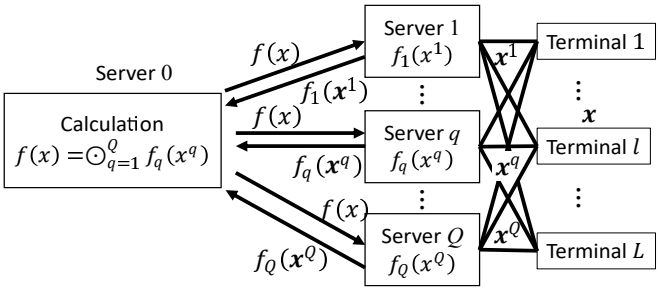

Fig. 1. An example of the proposed system : The data given from each terminal is divided into Q pieces and each piece is sent to a server. At Server q, $f_q(x^q)$ is calculated and sent to Server 0. On Server 0, the function $f(x)$ is calculated by using $f_q(x^q)$ for $q \in \{1, \cdots, Q\}$.

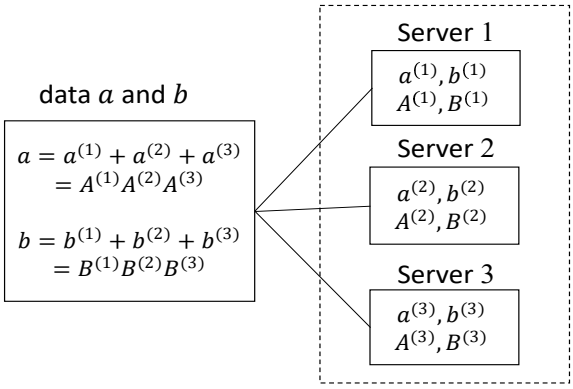

Fig. 2. The representation of secure divided data.

By using them, we can calculate four arithmetic operations of addition, subtraction, multiplication, and division without decrypting a and b as follows [7].
1)$a+b = (\sum_{q=1}^{Q} a^{(q)}) + (\sum_{q=1}^{Q} b^{(q)}) = \sum_{q=1}^{Q}(a^{(q)} + b^{(q)})$
2)$a-b = (\sum_{q=1}^{Q} a^{(q)}) - (\sum_{q=1}^{Q} b^{(q)}) = \sum_{q=1}^{Q}(a^{(q)} - b^{(q)})$
3)$ab = (\Pi_{q=1}^{Q} A^{(q)})(\Pi_{q=1}^{Q} B^{(q)}) = \Pi_{q=1}^{Q}(A^{(q)}B^{(q)})$
4)$a/b = (\Pi_{q=1}^{Q} A^{(q)})/(\Pi_{q=1}^{Q} B^{(q)}) = \Pi_{q=1}^{Q}(A^{(q)}/B^{(q)})$

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

Let $Q = 3$, $a = 6$ and $b = 8$. We divide a and b as $6 = 4 + (-1) + 3 = (-1)\times 2\times(-3)$ and $8 = 3 + 2 + 3 = 2\times(-4)\times(-1)$. In this case, we have $a^{(1)} = 4$, $a^{(2)} = -1$, $a^{(3)} = 3$, $A^{(1)} = -1$, $A^{(2)} = 2$, $A^{(3)} = -3$, $b^{(1)} = 3$, $b^{(2)} = 2$, $b^{(3)} = 3$, $B^{(1)} = 2$, $B^{(2)} = -4$ and $B^{(3)} = -1$, respectively. We calculate $a + b$ as follows (See Table I).

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

Further, we calculate $a\times b$ as follows (See Table II).

$$a\times b = (A^{(1)}\times B^{(1)})(A^{(2)}\times B^{(2)})(A^{(3)}\times B^{(3)})$$

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

	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

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

	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

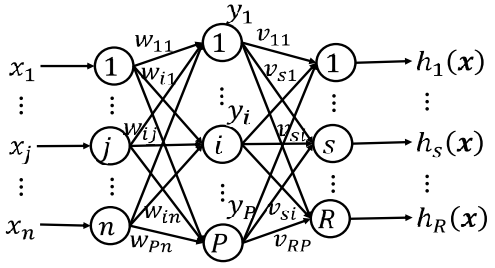

Fig. 3. An example of three layered Neural Network

$$= ((-1)\times 2)(2\times(-4))((-3)\times(-1)) = 48 \square$$

C. Neural Network and BP method

In this section, we explain the conventional three-layered Neural Network (NN) as shown in Fig.3 and BP method [12]. We determine the function $\boldsymbol{h} : J_{in}^{n} \to J_{out}^{R}$ for each input $\boldsymbol{x} \in J_{in}^{n}$ as follows. We have $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \cdots, h_R(\boldsymbol{x}))$, where $J_{in} = [0,1]$ or $[-1,1]$, $J_{out} = \{0,1\}$. In this case, we use the set of learning data, $X = \{(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l)) | \boldsymbol{x}^l \in J_{in}^{n}, \boldsymbol{d}(\boldsymbol{x}^l) \in J_{out}^{R}, l \in Z_L\}$, to determine the weights of NN, 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 $\boldsymbol{x}^l$.

We denote $W = \{w_{ij} | i \in Z_P, j \in Z_n^*\}$ and $V = \{v_{si} | s \in Z_R, i \in Z_P^*\}$ as the sets of weights. In this case, we can also calculate an output of NN 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, we use the Mean Square Error (MSE) for the learning data as the evaluation function for the BP method. In this case, we define the evaluation function 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, we determine each weight of W and V as follows [12]. 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) [12]
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 $V = \{v_{si}|s{\in}Z_R, i{\in}Z_P^*\}$ of weights
[**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), we calculate $y_i(\boldsymbol{x}^l)$ and $h_s(\boldsymbol{x}^l)$.
[**S**tep 3]
By using the following equations, we update $w_{ij}{\in}W$ and $v_{si}{\in}V$.

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

[**S**tep 4]
By using Eq.(3), we calculate the evaluation value $E(X, W, V)$.
[**S**tep 5]
If $E < \theta$ or $t > T$, the algorithm terminates else go to Step 2 with $t{\leftarrow}t + 1$.

□

D. NG and k-means methods

In this section, we explain the NG method, which is an unsupervised learning method based on the SDM [13].

Vector quantization techniques encode a data space, e.g., a subspace $X{\subseteq}R^d$, utilizing only a finite set $W = \{\boldsymbol{w}_i|i{\in}Z_r\}$ of reference vectors, where $|X| = L$, d and r are positive integers.

We denote $e_i(\boldsymbol{x}) \in Z_{r-1}^*$ as the neighborhood rank of $\boldsymbol{w}_i$ in W with respect to closeness to $\boldsymbol{x}$. That is, $\boldsymbol{w}_i$ is the $(e_i(\boldsymbol{x}) + 1)$-th nearest vector to $\boldsymbol{x}$ in W. Each parameter $\boldsymbol{w}_i \in W$ is updated by the following $\Delta\boldsymbol{w}_i$.

$$\Delta\boldsymbol{w}_i = \varepsilon{\cdot}\exp(-e_i(\boldsymbol{x})/\lambda){\cdot}(\boldsymbol{x} - \boldsymbol{w}_i), \qquad (6)$$

where $\varepsilon{\in}[0, 1]$ and λ is a positive real number. Eq.(6) means that the closer an element of W is to the input $\boldsymbol{x}$, the closer it is to $\boldsymbol{x}$.

For NG, the approximation by W of X is achieved by solving the minimization problem of the evaluation function E as follows.

$$E = \frac{1}{Lr} \sum_{\boldsymbol{x}^{(l)} \in X} \sum_{i=1}^{r} \frac{\exp(-e_i(\boldsymbol{x})/\lambda)}{\sum_{i'=1}^{r} \exp(-e_{i'}(\boldsymbol{x})/\lambda)} ||\boldsymbol{x}^{(l)} - \boldsymbol{w}_i||^2 \qquad (7)$$

If $\lambda{\rightarrow}0$, the method becomes the k-means method, which means that only the elements of W closest to the input $\boldsymbol{x}$ are brought close to $\boldsymbol{x}$.

The NG method is shown as follows, where T denotes the maximum number of learning times.
[**N**G method] NG(X, W, T) [13]
Input : The set $X = \{\boldsymbol{x}^l{\in}R^n|l{\in}Z_L\}$ of learning data
Output : The set $W = \{\boldsymbol{w}_i|i{\in}Z_r\}$ of reference vectors
[**S**tep 1]
Initialize W. Set $t{\leftarrow}0$.
[**S**tep 2]
Select a learning data $\boldsymbol{x}^l(l{\in}Z_L)$ randomly. We calculate the distance between $\boldsymbol{x}^l$ and $\boldsymbol{w}_i$ $(i{\in}Z_r)$, and calculate the neighborhood rank $e_i(\boldsymbol{x}^l)$ for $\boldsymbol{w}_i{\in}W$.
[**S**tep 3]
Update $\boldsymbol{w}_i{\in}W$ using Eq.(6).
[**S**tep 4]
If $t > T$, the algorithm terminates else go to Step 2 with $t{\leftarrow}t + 1$. □

The NG method includes the k-means method as a special case.

E. PSO and HPSO

Particle Swarm Optimization (PSO) is a method to solve the optimization problem using a global search method of solutions. First, we explain PSO [9]. We find the value of the determinant $\boldsymbol{x}$ that maximizes the nonlinear objective function $g(\boldsymbol{x})$ using PSO. To solve this problem, we prepare m individuals $P = \{P_k|k{\in}Z_m\}$ and perform a multipoint search that each solution is found in each individual. Let m be the number of individuals. We assign a decision variable $\boldsymbol{x}_k$ (solution candidate) to each individual P_k. In the solution search processes, we update 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 $\boldsymbol{p}_k$, we can expect to get an excellent solution with a large objective function value.

The following is the algorithm of PSO for learning of NN. The purpose of the algorithm is to determine the sets W and V of weights so that the MSE E for the learning data X is minimized. In the following, the maximum number T of learning times, threshold θ, and constant values c_0, c_1, and c_2 are set 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). We determine the function $E(X, W^k, V^k)$ as the MSE for learning data X when the weights W^k and V^k were used.
[**A**lgorithm PSO] [9], [10]
Input : The set X of learning data
Output : The sets W and V of weights
[**S**tep 1]
Let $\{W^k, V^k\}$ be the initial sets of W and V. We set $t{\leftarrow}0$, $W^{k^*}{\leftarrow}W^k$, $V^{k^*}{\leftarrow}V^k$ and $K{\leftarrow}\arg\ \min_{k^*}\{E(X, W^{k^*}, V^{k^*})\}$ of the initial values.
[**S**tep 2]
We select the numbers r_1 and r_2 randomly.
[**S**tep 3]
For $k{\in}Z_m$, we update $w_{ij}^k{\in}W^k$ and $v_i^k{\in}V^k$ as follows.

$$\begin{aligned} \triangle w_{ij}^k &\leftarrow c_0\triangle w_{ij}^k + c_1 r_1(w_{ij}^{k^*} - w_{ij}^k) + c_2 r_2(w_{ij}^K - w_{ij}^k) \\ w_{ij}^k &\leftarrow w_{ij}^k + \triangle w_{ij}^k \qquad (8) \\ \triangle v_{si}^k &\leftarrow c_0\triangle v_{si}^k + c_1 r_1(v_{si}^{k^*} - v_{si}^k) + c_2 r_2(v_{si}^K - v_{si}^k) \\ v_{si}^k &\leftarrow v_{si}^k + \triangle v_{si}^k \qquad (9) \end{aligned}$$

[**S**tep 4]
For $k{\in}Z_m$, if $E(X, W^k, V^k) < E(X, W^{k^*}, V^{k^*})$, then we set $W^{k^*}{\leftarrow}W^k$ and $V^{k^*}{\leftarrow}V^k$.
[**S**tep 5]

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

[Step 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, we update each weight of $\{W^k, V^k\}$ 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, we update the self-best weight $\{W^{k^*}, V^{k^*}\}$ in the past search for the solution $\{W^k, V^k\}$. In Step 5, we update the past best (optimal) solution $\{W^K, V^K\}$ in the whole.

On the other hand, it is known that PSO needs much time compared to the Steepest Descent method (SDM) like the BP and NG methods. To improve it, Hybrid Particle Swarm Optimization (HPSO) is proposed [10]. As HPSO uses both PSO and SDM, the optimal solution is calculated efficiently and needs a short time compared to the usual PSO.

First, we obtain HPSO for BP by replacing Step 3 of Algorithm PSO with the following Step 3'.

[Step 3']

For $k \in Z_m$,

We update W^k, V^k, W^{k*} and V^{k*} by using Eqs.(8) and (9).

We perform $BP^*(X, W^k, V^k, T_{BP})$ and $BP^*(X, W^{k*}, V^{k*}, T_{BP})$.

In Step 3', we mean $BP^*(X, W^k, V^k, T_{BP})$ and $BP^*(X, W^{k*}, V^{k*}, T_{BP})$ as 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.

Further, for the case of NG, we can use the following Step 3" instead of Step 3.

[Step 3"]

For $k \in Z_m$,

We update W^k and W^{k*} by using Eqs.(8).

We perform $NG^*(X, W^k, T_{NG})$ and $NG^*(X, W^{k*}, T_{NG})$.

In Step 3", we mean $NG^*(X, W^k, T_{NG})$ and $NG^*(X, W^{k*}, T_{NG})$ as the method $NG(X, W, T)$ with $W = W^k$ and $T = T_{NG}$, and $W = W^{k*}$ and $T = T_{NG}$, respectively.

III. Securely distributed system with divided data for HPSO

A. Data structure for the proposed method

In this section, we divide any learning data $(\boldsymbol{x}^l, \boldsymbol{d}(\boldsymbol{x}^l)) \in X$ with Q pieces as Eqs.(10) and (11) and they are stored in each server.

$$x_j^l = \Pi_{q=1}^Q x_j^{l(q)} \tag{10}$$

$$d_s(\boldsymbol{x}^l) = \sum_{q=1}^Q d_s^{(q)}(\boldsymbol{x}^l) \tag{11}$$

Likewise, we divide each weight of the sets W and V with Q pieces as Eqs.(12) and (13) and they are stored in each server.

$$w_{ij} = \Pi_{q=1}^Q w_{ij}^{(q)} \tag{12}$$

$$v_{si} = \Pi_{q=1}^Q v_{si}^{(q)} \tag{13}$$

We denote $W^{(q)} = \{w_{ij}^{(q)} | i \in Z_P, j \in Z_n^*\}$ and $V^{(q)} = \{v_{si}^{(q)} | s \in Z_S, i \in Z_P^*\}$. Let $\Pi_{q=1}^Q x_0^{(q)} = 1$.

First, we introduce the data structure to BP method. We calculate an output of the NN for the divided data and weights instead of Eqs.(1) and (2).

$$y_i(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\left(\sum_{j=0}^n \Pi_{q=1}^Q w_{ij}^{(q)} x_j^{(q)}\right)\right))} \tag{14}$$

Further, we divide the output y_i of the second layer to $y_i = \Pi_{q=1}^Q y_i^{(q)} (i \in Z_P^*)$ and each $y_i^{(q)}$ is sent to Server q. Let $\Pi_{q=1}^Q y_0^{(q)} = 1$. We calculate an output $h_s(\boldsymbol{x})$ of NN in Server 0, where $s \in Z_R$.

$$h_s(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\left(\sum_{i=0}^P \Pi_{q=1}^Q v_{si}^{(q)} y_i^{(q)}\right)\right)} \tag{15}$$

We divide the output $h_s(\boldsymbol{x})$ to $h_s(\boldsymbol{x}) = \sum_{q=1}^Q h_s^{(q)}(\boldsymbol{x})$ and each $h_s^{(q)}(\boldsymbol{x})$ is sent to Server q.

In this case, we calculate Mean Square Error (MSE) for the set X of learning data as follows.

$$E(X) = \frac{1}{2L} \sum_{l=1}^L \sum_{s=1}^R \left(\sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) \right)^2 \tag{16}$$

We update each of weights $w_{ij}^{(q)} (i \in Z_P, j \in Z_P^*)$ and $v_{si}^{(q)} (s \in Z_S, i \in Z_P^*)$ based on BP instead of Eqs.(4) and (5) as follows [7], [8].

$$\begin{aligned} w_{ij}^{(q)} = w_{ij}^{(q)} + \quad & \alpha \quad \sum_{s=1}^R \sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l))(1 - h_s^{(q)}(\boldsymbol{x}^l)) \\ & \times \quad \Pi_{q=1}^Q v_{si}^{(q)} h_s^{(q)}(\boldsymbol{x}^l)(1 - h_s(\boldsymbol{x}^l)) \\ & \times (\Pi_{q=1}^Q w_{ij}^{(q)} x_j^{l(q)}) / w_{ij}^{(q)} \end{aligned} \tag{17}$$

$$\begin{aligned} v_{si}^{(q)} = v_{si}^{(q)} + \alpha & \sum_{q=1}^Q (d_s^{(q)}(\boldsymbol{x}^l) - h_s^{(q)}(\boldsymbol{x}^l)) h_s(\boldsymbol{x}^l) \\ & \times (1 - h_s(\boldsymbol{x}^l))(\Pi_{q=1}^Q y_i^{(q)} v_{si}^{(q)}) / v_{si}^{(q)}) \end{aligned} \tag{18}$$

Tables III and IV show the relation among weights of HPSO and multiple servers. In Tables III and IV, we show 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, the Table IV shows that any element w^k of W^k for a solution candidate $\boldsymbol{x}_k$ is divided as $w^k = \sum_{q=1}^Q w^{k(q)}$ and each element $w^{k(q)}$ is stored in Server q.

In Table V, we show the conventional algorithm of BP [8]. In Table V, we first store each component of divided data and weights in a server. In Step 3, we calculate the product of the l-th data and the weight for the first layer and send them 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, we calculate the difference between the target

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	$\{W^m, V^m\}$	$\{W^{m*}, V^{m*}\}$	

and the output of NN in each server and they are integrated at Server 0 to calculate the updated amount of the weight. Further, the results are sent to each server. In Step 9, we update the weight of each server using the update amount. In Steps 11 and 12, the algorithm termination condition is checked.

Next, we introduce the data structure to NG method. To realize NG, we need 1) to determine the neighborhood rank of each element $\boldsymbol{w}_i$ of the set W with respect to the input data $\boldsymbol{x}^l$, and 2) to update all elements $\boldsymbol{w}_i$ of the set W. We realize the two steps of updating $\boldsymbol{w}_i$ using the distributed processing method. The initial condition is that each element of the divided data of input $\boldsymbol{x}^l$ and reference vector $\boldsymbol{w}_i$ is stored in each server, where $\boldsymbol{x}^l = (x_1^l, \cdots, x_j^l, \cdots, x_n^l)$, $x_j^l = \sum_{q=1}^{Q} x_j^{l(q)}$, and $\boldsymbol{w}_i = (w_{i1}, \cdots, w_{ij}, \cdots, w_{in})$, $w_{ij} = \sum_{q=1}^{Q} w_{ij}^{(q)}$.

[**T**he outline of the NG for SDCD] [8]
Input : The set $X = \{\boldsymbol{x}^l \in R^n | l \in Z_L\}$ of learning data
Output : The set $W = \{\boldsymbol{w}_i \in R^n | i \in Z_r\}$ of reference vectors
[**S**tep 1]

Calculate the neighborhood rank $e_i(\boldsymbol{x})$ for the input data $\boldsymbol{x}^l$ and the reference vector $\boldsymbol{w}_i \in W$. To achieve this, we calculate the distance D between input data and reference vector in each server and integrate them in Server 0 as follows.

$$D_{ijq}^{(l)} = \left(x_j^{l(q)} - w_{ij}^{(q)}\right) \quad (j = 1, \cdots, n, q = 1, \cdots, Q). \tag{19}$$

In Server 0, we calculate the distance between $\boldsymbol{x}^l$ and $\boldsymbol{w}_i$ as follows.

$$||\boldsymbol{x}^{(l)} - \boldsymbol{w}_i||^2 = \sum_{j=1}^{n} \left(\sum_{q=1}^{Q} D_{ijq}^{(l)} \right)^2 \tag{20}$$

[**S**tep 2]

We update each element $\boldsymbol{w}_i$ of W for $\boldsymbol{x}^l$ using the following formula. In this case, because we use both $\boldsymbol{w}^l$ and $\boldsymbol{w}_i$ of an additive form in the data division, and have the relation of $\frac{\partial w_{ij}}{\partial w_{ij}^{(q)}} = 1$, and $\frac{\partial}{\partial w_{ij}^{(q)}} \sum_{q=1}^{Q} w_{ij}^{(q)} = 1$.

Therefore, we have

$$\begin{aligned} \Delta w_{ij}^{(q)} &= \frac{\partial E}{\partial w_{ij}^{(q)}} = \frac{\partial E}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial w_{ij}^{(q)}} \\ &= \varepsilon \cdot \exp(-e_i(\boldsymbol{x}^l)/\lambda) \cdot (x_j^l - w_{ij}) \end{aligned} \tag{21}$$

[**S**tep 3]

Repeat Steps 1) and 2) if learning completion conditions is not satisfied. □

Table VI shows the detailed algorithm of the NG for SDCD [8]. The maximum number T_{max} of learning times is given in advance. The data and reference vectors were divided and stored in each server. In Step 1, the set l of a natural number is selected randomly. In Step 2, we calculate $D_{ij}^{l(q)}$ in each server and send it to Server 0. In Step 3, we calculate the update amount $\Delta w_{ij}^{(q)}$ of reference vector $w_{ij}^{(q)}$ by integrating $D_{ij}^{l(q)}$ in Server 0 and send it to each server. In Step 4, the reference vectors are updated. In Step 5, if the learning time t arrives at T, the algorithm terminates.

B. The proposed HPSO methods for BP and NG

First, we explain the proposed HPSO for BP. In the case of PSO, we update each piece of $w_{ij}^{k(q)}$ and $v_{si}^{k(q)}$ of divided weights by using Eqs.(22) and (23) instead of Eqs.(8) and (9).

$$\begin{aligned} \triangle w_{ij}^{k(q)} &= c_0 \triangle w_{ij}^{k(q)} + c_1 r_1 (w_{ij}^{k^*(q)} - w_{ij}^{k(q)}) \\ &\quad + c_2 r_2 (w_{ij}^{K(q)} - w_{ij}^{k(q)}) \\ w_{ij}^{k(q)} &= w_{ij}^{k(q)} + \triangle w_{ij}^{k(q)} (j \in Z_n^*, i \in Z_P) \end{aligned} \tag{22}$$

$$\begin{aligned} \triangle v_{si}^{k(q)} &= c_0 \triangle v_{si}^{k(q)} + c_1 r_1 (v_{si}^{k^*(q)} - v_{si}^{k(q)}) \\ &\quad + c_2 r_2 (v_{si}^{K(q)} - v_{si}^{k(q)}) \\ v_{si}^{k(q)} &= v_{si}^{k(q)} + \triangle v_{si}^{k(q)} (s \in Z_R, i \in Z_P^*) \end{aligned} \tag{23}$$

We denote $W^{k(q)} = \{w_{ij}^{k(q)} | i \in Z_P, j \in Z_n^*\}$ and $V^{k(q)} = \{v_{si}^{k(q)} | s \in Z_S, i \in Z_P^*\}$ for the sets $\{W^k, V^k\}$.

The general flow of HPSO for BP is shown as follows.

[**T**he general flow of the proposed HPSO for BP]
Input : The set $X = \{x_j^{l(q)}, d_s^{(q)}(\boldsymbol{x}^l) | l \in Z_L, j \in Z_n, q \in Z_Q\}$
Output : The set of best in whole $\{W^{K(q)}, V^{K(q)} | q \in Z_Q\}$
T : Maximum number of learning epochs for PSO
T_{BP} : Maximum number of learning epochs for BP
1 : Initialize the sets $\{W^{k(q)}, V^{k(q)} | k \in Z_m, q \in Z_Q\}$.
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.(22) and (23)
  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.(17) and (18)
      3.3.2.2 : $t_{BP} \leftarrow t_{BP} + 1$
      End
    End
  3.3.3 : $t \leftarrow t + 1$
End

□

We show the proposed HPSO for BP in Table VII. In Table VII, $E(X, \{W^{k(q)}, V^{k(q)}\})$ means the calculation result of Eq.(16) when $\{W^{k(q)}, V^{k(q)}\}$ is used for $\{W^{(q)}, V^{(q)}\}$.

Server q has the set of q-th pieces of W^k and V^k in advance. In Steps 1 and 2, we set the self-best solution candidates and the overall best solution. In Steps 3 and 4, we update $w_{ij}^{k(q)}$ and $v_{si}^{k(q)}$ based on the update formula of PSO. In Step 5, we update each weight for m NNs based on

TABLE IV
RELATION AMONG WEIGHTS FOR DIVIDED SOLUTION CANDIDATE WITH EACH SERVER : THE TABLE SHOWS WHICH SOLUTIONS OF PSO CONSIST OF WHICH WEIGHT PIECES OF SERVERS. FOR EXAMPLE, A SOLUTION CALCULATE x_k IS CONSISTED OF THE SET $\{W^{k(q)}, V^{k(q)}\}$ FOR $q \in Z_Q$.

Server	The set of partitions of divided solution candidate	Solution candidate $\boldsymbol{x}_k$	Self-best $\boldsymbol{p}_k$	Best in whole $\boldsymbol{a}$
Server 1	$\{W^{k(1)}, V^{k(1)} \mid k \in Z_m\}$	○		
	$\{W^{k*(1)}, V^{k*(1)} \mid k \in Z_m\}$		○	
	$\{W^{K(1)}, V^{K(1)}\}$			○
⋮				
Server q	$\{W^{k(q)}, V^{k(q)} \mid k \in Z_m\}$	○		
	$\{W^{k*(q)}, V^{k*(q)} \mid k \in Z_m\}$		○	
	$\{W^{K(q)}, V^{K(q)}\}$			○
⋮				
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 BP method in Table V. In Steps 6 and 7, we update self-best $\{W^{k*(q)}, V^{k*(q)}\}$. In Step 8, we update the overall best $\{W^{K(q)}, V^{K(q)}\}$. In Step 9, the final condition is checked.

Next, we explain the proposed HPSO for NG. In Table VIII, we show the proposed HPSO for NG. In Table VIII, we define $E(X, \{W^{k(q)}\})$ as the calculation result of Eq.(7) when $\{W^{k(q)}\}$ is used for $\{W^{(q)}\}$.

Server q has the set of q-th pieces of W^k in advance. In Steps 1 and 2, we set the self-best solution candidates and the overall best solution. In Steps 3 and 4, we update $w_{ij}^{k(q)}$ based on the update formula of PSO. In Step 5, we update each weight for m sets of reference vectors based on the NG method in Table VI. In Steps 6 and 7, we update self-best $\{W^{k*(q)}\}$. In Step 8, we update the overall best $\{W^{K(q)}\}$. In Step 9, the final condition is checked.

IV. NUMERICAL SIMULATIONS FOR THE PROPOSED METHODS

In this section, we perform numerical simulations for function approximation and pattern classification. The BP (or NG), HPSO, and BP (or NG) for SDCD mean the conventional BP (or NG), HPSO methods, and the conventional BP (or NG) for SDCD, respectively. The Proposed means the proposed HPSO for BP or NG.

A. Function Approximation

This simulation uses four systems specified by the following functions determined for x_1, $x_2 \in [0, 1]$ and $y \in [0, 1]$.

$$y = \sin(\pi x_1^3) x_2 \tag{24}$$

$$y = \frac{\sin(2\pi x_1^3)\cos(\pi x_2) + 1}{2} \tag{25}$$

$$y = \frac{1.9(1.35 + \exp(x_1)\sin(13(x_1 - 0.6)^2)}{7} \times \exp(-x_2)\sin(7x_2)) \tag{26}$$

$$y = \frac{\sin(10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2) + 1}{2} \tag{27}$$

The numbers of learning and test data are 100, respectively. Each learning 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.

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

B. Pattern classification for the proposed BP

We perform numerical simulation for pattern classification using benchmark problems of Iris, Wine, Sonar, and BCW in the UCI database[14]. We show the details in Table X. In Table X, #data, #input, and #class mean the numbers of data, input, and class for each data, respectively. In the simulation, we use 5-fold cross-validation as the evaluation method. 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 the proposed method, respectively. The threshold is $\theta = 0.03$ for Iris and Wine and $\theta = 0.04$ for Sonar and BCW.

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

C. Pattern clustering for the proposed NG

In this section, we perform pattern clustering using NG (and k-means as special case) method which is one of

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.(14). 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.(15). 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.(16).	
Step 11	If $E < \theta$ or $t < T$, algorithm terminates. Otherwise go to Step 2 with $t \leftarrow t + 1$.	

TABLE VI
ALGORITHM OF THE PROPOSED NG METHOD.

	Server 0	Server q
Initialize	Determine the values ε_{int}, ε_{fin}. Set $t = 1$.	Store $\{x_{jq}^{(l)} \mid l \in Z_L, j \in Z_n\}$. Initialize $\{w_{ij}^{(q)} \mid i \in Z_r, j \in Z_n\}$.
Step 1	Select a natural number $l \in Z_L$ randomly and send to each server.	
Step 2		Calculate $D_{ij}^{l(q)} = (x_j^{l(q)} - w_{ij}^{(q)})$ $(i \in Z_r, j \in Z_n)$ and send to Server 0.
Step 3	Based on Eq.(20), calculate $\exp(-e_i(\boldsymbol{x}^l)/\lambda)$ and $\Delta w_{ij} = \varepsilon_{int} \left(\frac{\varepsilon_{fin}}{\varepsilon_{int}}\right)^{\frac{t}{T}} \exp(-e_i(\boldsymbol{x}^l)/\lambda) \sum_{q=1}^{Q} D_{ij}^{l(q)}$ and send to each server.	
Step 4		Update $\{w_{ij}^{(q)} \mid i \in Z_r, j \in Z_n\}$ as follows. $w_{ij}^{(q)} \leftarrow w_{ij}^{(q)} + \Delta w_{ij}$
Step 5	If $t = T_{max}$, the algorithm terminates. Otherwise, Set $t \leftarrow t + 1$ and go to Step 2.	

unsupervised learning. The NG (k-means), HPSO, and NG (k-means) for SDCD mean the conventional NG (k-means), HPSO, and the conventional NG (k-means) for SDCD, respectively. The Proposed means the proposed HPSO for NG (k-means). We perform clustering the four benchmark datasets, Iris, Wine, Sonar and BCW [14].

The conditions of clustering simulation for NG and k-means are as follows. The number r of reference vectors is 3 in the case of Iris and Wine and 2 in the case of Sonar and BCW. In the proposed method, let $Q = 3$. The maximum numbers of learning are shown in Table XII.

After learning, we compare the conventional and proposed NG (or k-means) methods in terms of the global purity (GP) of Eq.(28) and the evaluation function of Eq.(7), where $n_{i,j}$ is the number of data belonging to the i-th cluster and the j-th actual class.

$$GP = \frac{1}{L} \sum_{i \in Z_r} \max_{j \in Z_r} (n_{i,j}) \times 100(\%) \tag{28}$$

The GP and the evaluation function values (accuracy) for the NG method are shown in Table XIII. Likewise, the GP and the evaluation function values (accuracy) for the k-means method are shown in Table XIV. The results in the Tables XIII and XIV are the average of 20 trials each.

In Tables XIII and XIV, the proposed methods show almost the same accuracy compared to the conventional HPSO.

V. CONCLUSION

In this paper, we proposed a secure divided learning method of HPSO for BP and NG which are supervised and

TABLE VII
PROPOSED HPSO FOR SDCD (BP)

	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.(22) and (23), 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.(16), $E(X,\{W^{k(q)}, V^{k(q)}\})$ 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 VIII
PROPOSED HPSO FOR SDCD (NG)

	Server 0	Server$q$ $(1\leq q\leq Q)$
Initialize		Initialize $\{W^{k(q)} \mid k\in Z_m\}$
Step 1		$W^{k^*(q)}\leftarrow W^{k(q)}$ $(k\in Z_m)$
Step 2	Calculate $K\leftarrow \arg\min_{k^*}\{E(X,\{W^{k^*(q)})\}\}$ 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.(22), each of $\{w_{ij}^{k(q)} \mid i\in Z_r\}$ and is updated$(k\in Z_m)$.
Step 5	$NG(X, W^{k(q)}, T_{NG})$ and $NG(X, W^{k*(q)}, T_{NG})$ for $k\in Z_m$: Update $\{W^{k(q)}\}$ and $\{W^{k*(q)}\}$ for $k\in Z_m$ by using NG of Table VI.	
Step 6	By using Eq.(7), $E(X,\{W^{k(q)}\})$ and $E(X,\{W^{k^*(q)})\}$ are calculated. If $E(X,\{W^{k(q)}\}) < E(X,\{W^{k^*(q)}\})$, then go to Step 7. Otherwise go to Step 8.	
Step 7		Set $W^{k^*(q)}\leftarrow W^{k(q)}$ $(k\in Z_m)$.
Step 8	Calculate $K\leftarrow \arg\min_{k^*}\{E(X, W^{k^*})\}$ and send it to each server.	
Step 9	If or $t > T$, then the algorithm terminates. Otherwise, go to Step 3 with $t\leftarrow t+1$.	

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

		Eq.(24)	Eq.(25)	Eq.(26)	Eq.(27)
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 XI
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

TABLE X
THE DATASET FOR PATTERN CLASSIFICATION

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

TABLE XII
THE SIMULATION CONDITIONS FOR NG AND K-MEANS METHODS

		Iris	Wine	Sonar	BCW
NG	T	15000	18000	21000	70000
HPSO	T	10	10	10	10
	T_I	1500	1800	2100	7000
NG for SDCD	T	15000	18000	21000	70000
Proposed for HPSO	T	10	10	10	10
	T_I	1500	1800	2100	7000

unsupervised methods. Then, HPSO is a hybrid PSO that combines PSO, which is a global search, and BP or NG, which is a local search method. The feature of the proposed method is a learning method using global and local search to find the optimal solution, and it does not use learning data and parameters themselves but uses a distributed processing with divided data and parameters. Numerical simulation results show that the proposed method is almost as accurate as the conventional HPSO in both supervised and unsupervised learning. In the future, we plan to propose other learning methods that perform a global search with secure distributed

TABLE XIII
SIMULATION RESULTS FOR CONVENTIONAL AND PROPOSED NG METHODS.

		Iris	Wine	Sonar	BCW
NG	GP(%)	4.1	7.0	45.0	3.7
	MSE	0.006449	0.047623	0.691680	0.145834
HPSO	GP(%)	4.1	6.9	45.2	3.7
	MSE	0.006421	0.047698	0.692082	0.146119
NG for SDCD	GP(%)	4.0	7.0	45.1	3.6
	MSE	0.006432	0.047729	0.693154	0.146063
Proposed for HPSO	GP(%)	4.0	6.7	45.3	3.8
	MSE	0.006344	0.047155	0.682860	0.143236

TABLE XIV
SIMULATION RESULTS FOR CONVENTIONAL AND PROPOSED K-MEANS METHODS.

		Iris	Wine	Sonar	BCW
k-means	GP(%)	4.9	7.4	45.6	4.0
	MSE	0.019855	0.146691	1.415555	0.296711
HPSO	GP(%)	4.4	7.4	44.3	4.0
	MSE	0.019800	0.146310	1.431822	0.297036
k-means for SDCD	GP(%)	4.6	7.6	45.6	4.2
	MSE	0.019779	0.146762	1.418258	0.296943
Proposed for HPSO	GP(%)	4.0	6.9	45.3	3.9
	MSE	0.019121	0.143253	1.383543	0.288883

processing using decomposed data.

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] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," arXiv:1610.05492, 2017.

[6] 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).

[7] 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.

[8] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, N. Shiratori, "Machine Learning with Distributed Processing using Secure Divided Data: Towards Privacy-Preserving Advanced AI Processing in a Super-Smart Society," *Journal of Networking and Network Applications*, vol.2, issue 1, pp.48-60, 2022.

[9] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proc. IEEE Int. Conf. Neural Networks*, pp.1942-1948, 1995.

[10] 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.

[11] H. Miyajima, N. Shigei, H. Miyajima, N. Shiratori. "Securely Distributed Computation with Divided Data for Particle Swarm Optimization," Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2021, 20-22 October, 2021, Hong Kong, pp1-6.

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

[13] 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.

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