Machine Learning with Distributed Processing using Secure Divided Data

Towards Privacy-Preserving Advanced AI Processing in a Super-Smart Society

Hirofumi Miyajima[1], Noritaka Shigei[2*], Hiromi Miyajima[2], and Norio Shiratori[3]
[1]Nagasaki University, 1-14 Bunkyomachi, Nagasaki city, Nagasaki 852-8521, Japan
[2]Kagoshima University, 1-21-40, Korimoto, Kagoshima, 890-0065, Japan
[3]Chuo University, 1-13-27, Kasuga, Bunkyoku, Tokyo, 112-8551, Japan
[*]Corresponding author

Towards the realization of a super-smart society, AI analysis methods that preserve the privacy of big data in cyberspace are being developed. From the viewpoint of developing machine learning as a secure and safe AI analysis method for users, many studies have been conducted in this field on 1) secure multiparty computation (SMC), 2) quasi-homomorphic encryption, and 3) federated learning, among other techniques. Previous studies have shown that both security and utility are essential for machine learning using confidential data. However, there is a trade-off between these two properties, and there are no known methods that satisfy both simultaneously at a high level.

In this paper, as a superior method in both privacy-preserving of data and utility, we propose a learning method based on distributed processing using simple, secure, divided data and parameters. In this method, individual data and parameters are divided into multiple pieces using random numbers in advance, and each piece is stored in each server. The learning of the proposed method is achieved by using these data and parameters as they are divided and by repeating partial computations on each server and integrated computations at the central server. The advantages of the proposed method are the preservation of data privacy by not restoring the data and parameters during learning; the improvement of usability by realizing a machine learning method based on distributed processing, as federated learning does; and almost no degradation in accuracy compared to conventional methods. Based on the proposed method, we propose backpropagation and neural gas (NG) algorithms as examples of supervised and unsupervised machine learning applications. Our numerical simulation shows that these algorithms can achieve accuracy comparable to conventional models.

Index Terms—**machine learning, secure divided data, distributed processing, federated learning, neural networks, and neural gas.**

I. Introduction

Considerable effort is being undertaken towards the realization of a sustainable society via the pursuit of Sustainable Development Goals (SDGs), which consist of 17 global goals and 169 success criteria [1], [2]. To promote a sustainable society from the industrial side in line with these SDGs as guidelines, Japan has established the theme of Society 5.0. In the super-smart society of the Society 5.0 vision, cyberspace and physical space (the real world) have become highly integrated, and artificial intelligence (AI) existing in cyberspace instantly finds relevant information according to emerging situations and provides the results of automated analysis to the real world [3], [4]. In Society 5.0, a massive amount of information from real-world sensors and IoT devices such as smartphones continually accumulates in cyberspace. In cyberspace, AI analyzes this big data to meet individual needs. As a result, useful information is quickly brought to the real world.

In such a super-smart society, while significant contributions to "comfortable living conditions," "promotion of public health," and "support for the independence of the elderly" are expected, "digital divide," "insufficient ability to explain information from AI," "invasion of privacy" and "strengthening systems of control over persons" are the demerits [5]–[7]. How, then do we construct a super-smart society while preventing privacy violations and maintaining personal autonomy? To solve this problem, the development of an AI analysis method that preserves the privacy of big data in cyberspace is required. In this study, this is referred to as advanced AI processing. Therefore, in this field, many studies have been conducted on advanced AI methods for performing machine learning while maintaining user safety and security [5]–[7].

The development of a technology that performs computational processing while maintaining data confidentiality is a design goal of big data processing architectures in cloud computing and edge network system, which comprise the infrastructure that will be leveraged to execute AI processing in the Society 5.0 model. From the definition of computer security as described in [8], data privacy is achieved through data confidentiality, while the fundamental goal of data security is to achieve the three goals of data confidentiality, integrity, and availability. Data privacy mechanisms are being actively developed for machine learning from a security point of view including privacy, and representative existing methods include, 1) secure multiparty computation (SMC) [9]–[12], 2) quasi-isomorphic encryption [13], [14], and 3) federated learning (FL) [15]–[17].

Method 1) first divides the data into multiple pieces using

Manuscript received December 11, 2021; revised April 1, 2022. Corresponding author: Noritaka Shigei (email: shigei@eee.kagoshima-u.ac.jp).

random numbers and stores each in each server. Then, using the data pieces of each server, cooperative calculations are performed between servers while maintaining the confidentiality of the data. Because the data are processed using only operations that preserve confidentiality while in a divided state, data confidentiality is preserved [9]–[12].

Method 2) first encrypts each data sample. Next, the desired calculation is performed using the encrypted data, and finally, the computation is completed by decrypting the final result. Therefore, the approach requires an encryption method that allows the calculation to be performed while the data are encrypted [13], [14].

Method 3) uses a single central server and multiple servers. In this method, the dataset is partitioned into multiple subsets, and each subset is stored on each server. Machine learning is performed independently on each server, and the results are sent to the central server. The central server integrates these, calculates the results for all data, and sends the results to each server. The calculation process for obtaining the partial result on the servers and the total result on the central server is repeated. In this process, each server uses only its own data for the calculation, and the data are not sent out from the server. Therefore, the confidentiality of the data is considered guaranteed [15]–[17]. FL has been applied to many applications such as mobile keyboard prediction [28], prediction using medical data [29], low-latency communication [30] and anomaly detection [31].

In these cases, Methods 1) and 2) strictly preserve data confidentiality by using data encryption and random numbers, and Method 3) partitions all data into subsets and distributes them to each server to distribute the data. This method executes learning by distributed processing without sending the data from each server. Each approach has advantages and disadvantages. Methods 1) and 2) are extremely confidential in terms of security. However, their utilization in machine learning is limited in application. For this reason, it has been studied to increase the utility of SMC. For example, methods with partially reduced communication and computation costs [19], [20] and a management framework to facilitate SMC deployment [32] have been proposed. Method 3) is highly utilizable for many machine learning problems owing to the simplicity of the procedure. It is also highly adaptable to edge computing for IoT. However, the security level is low compared to Methods 1) and 2) [27]. Therefore, many studies have been done to improve the security of FL, such as [22]–[26], [33]–[35]. From the viewpoint of ensuring confidentiality, cryptographic techniques and perturbation techniques are mainly used. For the former, hybrid methods with 1) and 2) have been proposed [21], [23]. The latter techniques are based on differential privacy [18] using perturbation or noise to the data, and it has been studied to improve the trade-off between accuracy and confidentiality [36], [37]. From the above, the goal of these approaches is to find a balanced method between the utility and security levels.

In this paper, we propose a learning method that performs distributed processing using simple, secure, divided data and parameters. In this method, individual data and parameters are divided into multiple pieces in advance using random numbers as in Method 1). Learning is achieved by iteratively performing independent calculations for each server and integrating the calculations on the central server. In addition, unlike Method 3), there is no need to train on the entire dataset in each server and no need to inform each server of the parameters resulting from training on all the data. Thus, it is possible to learn with high confidentiality by using divided data and parameters. Based on these characteristics, the proposed method can be expected to have a high utilization value and low security risk.

Machine learning aims to estimate the parameters that connect the relationships among data from a given set of source data. Machine learning methods are divided into two types, including supervised learning, which learns the relationship between input and output data, and unsupervised learning, which approximates the data distribution.

In this paper, as an example of supervised and unsupervised learning, we propose the BP and NG algorithms based on distributed processing using simple, secure, divided data and parameters. The remainder of this work is organized as follows. In Section II, we define the divided data of the additive and product types as simple, secure, divided data and parameters for the BP and NG methods. In Section III, we propose learning methods based on distributed processing using simple, secure, divided data distributed to each server. In section III-A, the method of parameter updating for the proposed methods is theoretically proven to be feasible for machine learning. In Section IV, we compare the accuracy of the conventional BP and NG methods with those of the proposed BP and NG methods by numerical experiments. Finally, in Section V, we summarize the contributions of this study to society and discuss future prospects.

II. Preliminaries

In this section, we first explain the concept of computational methods for distributed processing with divided data. Next, we explain the data representation used in the conventional and proposed methods. Furthermore, we introduce the steepest descent method and explain the BP and NG methods. Finally, we explain FL as an example of the conventional method using subsets of data and distributed processing by multiple servers.

A. Secure computation and configuration for cloud and edge systems

Let us present an outline of distributed processing. Fig.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 sample and a function, respectively. Each data point x is divided into pieces, and each piece is sent to each server. In the case of Fig.1, each data is divided into an addition form.

First, each data point x from each terminal is randomly divided into Q pieces 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 $f_q(\cdot)$ is 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 $\odot$ is an integrated operation. If the calculation result cannot

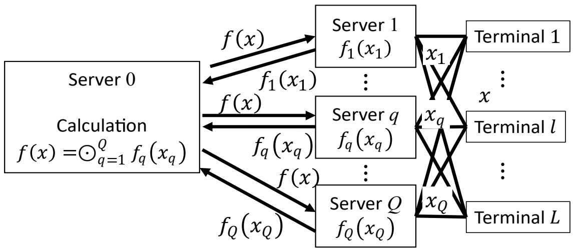

Fig. 1. An example of the proposed system

TABLE I
DATA STRUCTURE OF HPD

	ID	Subject A a	Subject B b
Server 1	1	50	80
	2	40	50
	3	65	30
Server 2	4	70	62
	5	80	40
	average	61	52.4

be obtained in one process, multiple processes are repeated. In these processes, each server q does not use data $x_{q'}$ of other server $q' \neq q$.

The problem is defined as that of determining the function f_q in each server and the operation $\odot$ in Server 0.

B. Data representation for the conventional and the proposed methods

In this section, horizontally partitioned data (HPD), which is data structures used in conventional method such as FL, is explained using an example, as shown in Table I [36]–[38], and data structure of the proposed method is introduced.

First, let us explain HPD. In Table I, the grades a and b of five subjects (Subject) A and B with IDs from 1 to 5 are given. The purpose of the calculation is to determine the average score for each subject. All the data are stored in two servers, Server 1 and Server 2, as follows.

Server 1: dataset for ID=1, 2, 3.

Server 2: dataset for ID=4, 5.

In this case, the average of data subsets A and B in Server 1 is calculated as $(50 + 40 + 65)/3$ and $(80 + 50 + 30)/3$, respectively. Similarly, the average of data subsets A and B in Server 2 is computed as $(70 + 80)/2$ and $(62 + 40)/2$, respectively. As a result, we obtain 61.0 and 52.4 as the averages of subsets A and B, respectively. In this case, each data point is not encrypted, but each server possesses only about half of the total data, which is more secure than the case in which all data are stored by a single server. In addition, as the number of servers increases, security increases.

Further, based on the data in Table I, we use Table II to describe the simple, secure, divided data used in this study [39], [40]. Unlike the conventional method, the proposed method stores data as divided data on each server. The number of servers storing the data was set to two, as shown in TableI.

We divide the real numbers a and b in the form of sum and product, $a = a_1 + a_2 = A_1A_2$ and $b = b_1 + b_2 = B_1B_2$, respectively. Here, a_1, a_2, A_1, A_2, b_1, b_2, B_1, and B_2 are real values chosen at random, a_1, b_1, A_1, A_1, B_1, A_1, and B_1 are stored in Server 1, and A_2, B_2, A_2, and B_2 are stored in Server 2. For example, the real numbers a and b are divided as follows.

$a = a_1 + a_2 : a_1 = a(r_1/10),\ a_2 = a(1 - r_1/10)$
$b = b_1 + b_2 : b_1 = b(r_1/10),\ b_2 = b(1 - r_1/10)$
$a = A_1A_2 : A_1 = \sqrt{a}(r_2/10),\ A_2 = \sqrt{a}(10/r_2)$
$b = B_1B_2 : B_1 = \sqrt{b}(r_2/10),\ B_2 = \sqrt{b}(10/r_2)$.

Let r_1 and r_2 be randomly chosen real numbers satisfying $-9 \le r_1 \le 9$, $r_1 \le 1$, $0.2 \le r_2 \le 9$, and $r_2 \le 1$, respectively.

For example, for data ID=1, $a_1 = 50\times(4/10) = 20$, $a_2 = 50\times(1 - 4/10) = 30$, $A_1 = \sqrt{50}\times(9/10) = 6.31$, $A_2 = \sqrt{50}\times(10/9) = 7.86$.

Let us explain the calculation of the sum and mean of subject A. In Server 1, the sum of items a_1 is obtained, and in Server 2, the sum of items a_2 is obtained. In this example, the sum of items a_1 is -23, and the sum of items a_2 is 328. By finding these sums, we can obtain the sum of a, $-23 + 328 = 305$. Similarly, by using the mean of -4.6 for a_1 and the mean of 65.6 for a_2, we can obtain the mean of a, $-4.6 + 65.6 = 61$. In this case, each data sample is divided using random numbers, and the server can perform the calculation by using each piece of data and integrating the results.

In general, using such an additive or product data decomposition method, the following relation holds.

1) $a + b = (a_1 + b_1) + (a_2 + b_2)$
2) $a - b = (a_1 - b_1) + (a_2 - b_2)$
3) $ab = (A_1B_1)(A_2B_2)$
4) $a/b = (A_1/B_1)(A_2/B_2)$

Using the above formulas, we can perform the four arithmetic operations on a and b using only randomly selected real numbers a_1, a_2, A_1, A_2, b_1, B_2, B_1, and B_2, without decoding a and b, and use the results of the computation on each server.

C. Steepest Descent Method

Machine learning aims to estimate the input-output relationship for a given learning data by estimating the parameters for a model. This section explains the steepest descent method (SDM) to estimate parameters [41], [42].

SDM is a method designed to find the parameter $\boldsymbol{\theta}$ that minimizes the objective function $J(\boldsymbol{\theta})$.

For the parameters, we repeatedly apply the following update equations to get close to the appropriate values using the gradient method.

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta \nabla J(\boldsymbol{\theta}), \tag{1}$$

where η is the learning coefficient, which is a real number that determines the step size of the update. In addition, $\nabla J(\boldsymbol{\theta})$ is the amount of update for the parameter $\boldsymbol{\theta}$.

The parameter $\boldsymbol{\theta}$ can be used to obtain a local solution of the function $J(\boldsymbol{\theta})$ by repeating Eq.(1).

If SDM is used for machine learning, three types of learning methods have been reported in the relevant literature [41]:

TABLE II
AN EXAMPLE OF SECURE COMPUTATION

ID	subject A	subject B	Additive form					Product form				
				a		b			A		B	
	a	b	r_1	a_1	a_2	b_1	b_2	r_2	A_1	A_2	B_1	B_2
1	50	80	4	20	30	32	48	9	6.31	7.86	8.05	9.94
2	40	50	-6	-24	64	-30	80	2	1.27	31.62	1.41	35.36
3	65	30	2	13	52	6	24	0.8	0.65	100.78	0.44	68.47
4	70	62	-8	-56	126	-49.6	111.6	5	4.18	16.73	3.94	15.75
5	80	40	3	24	56	12	28	4	3.58	22.36	2.53	15.81
sum	305	262		-23	328	-29.6	291.6					
average	61	52.4		-4.6	65.6	-5.92	58.32					

online learning, mini-batch learning, and batch learning, depending on how the objective function J in Eq.(1) is given. We explain them below.

For any natural number i, let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$. Let D be the set of learning data, and $|D| = L$. In addition, the set D is partitioned into $D = \bigcup_{l=1}^{N} B_l$ ($B_i \cap B_j = \emptyset$) and N subsets $B_1, \cdots, B_N$, and $|B_l| = b_l$ ($l \in Z_N$). In this case, $L = \sum_{l=1}^{N} b_l$. The learning method using SDM is as follows [41]. Let $E(X, \boldsymbol{\theta})$ be the error function of $\boldsymbol{\theta}$ for dataset X. First, we set $t = 1$.

[**S**tep 1] Select a natural number $l \in Z_N$ randomly and determine a subset B_l of the learning data to be used in updating the parameters.
[**S**tep 2] Update $\boldsymbol{\theta}$ using Eq.(1). Here, we use $E(B_l, \boldsymbol{\theta})$ instead of $J(\boldsymbol{\theta})$.
[**S**tep 3] Calculate $E(D, \boldsymbol{\theta})$ using the updated parameters $\boldsymbol{\theta}$. If $E(D, \boldsymbol{\theta})$ is sufficiently small, terminate the algorithm; else, return to Step 1 as $t \leftarrow t + 1$. □

This method is called online learning in the case of $N = L$, batch learning in the case of $N = 1$, and mini-batch learning in other cases.

Machine learning methods based on SDM include the BP method, k-means method, NG, self-organization map (SOM), and fuzzy modeling [41]–[43].

D. Neural Network and BP method

Let us explain the BP method, which is a supervised learning method based on SDM. Here, we describe BP learning for a three-layered neural network (NN) without loss of generality [40], [42].

Let $\boldsymbol{h} : J_{\text{in}}^n \to J_{\text{out}}^R$ be the map constructed by the NN, where $h(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \cdots, h_R(\boldsymbol{x}))$ for $\boldsymbol{x} \in J_{\text{in}}^n$. Let $J_{\text{in}} = [0, 1]$ or $[-1, 1]$ and $J_{\text{out}} = \{0, 1\}$ herein. In this case, the set of L learning data, $X = \{(\boldsymbol{x}^{(l)}, \boldsymbol{d}(\boldsymbol{x}^{(l)})) | \boldsymbol{x}^{(l)} \in J_{\text{in}}^n, \boldsymbol{d}(\boldsymbol{x}^{(l)}) \in J_{\text{out}}^R, l \in Z_L\}$, is used to determine the weights $W = \{w_{ij} | i \in Z_P, j \in Z_n^*\}$ and $V = \{v_{si} | s \in Z_R, i \in Z_P^*\}$, which are parameters of NN. Here, $\boldsymbol{d}(\boldsymbol{x}^{(l)}) = \left(d_1(\boldsymbol{x}^{(l)}), \ldots, d_R(\boldsymbol{x}^{(l)})\right)$ represents the output of the supervised data for input $\boldsymbol{x}^{(l)}$. In this case, an output of NN is calculated as follows.

$$y_i(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\left(\sum_{j=0}^{n} w_{ij} x_j\right)\right)}, \tag{2}$$

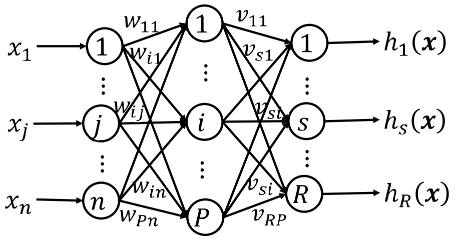

Fig. 2. An example of three-layered neural network

$$h_s(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\left(\sum_{i=0}^{P} v_{si} y_i(\boldsymbol{x})\right)\right)}. \tag{3}$$

where $x_0 = 1$, $y_0 = 1$, and w_{i0} and v_{s0} are thresholds for each layer.

To determine the weights, the mean square error (MSE) for the learning data was used as the evaluation function for the BP method. In this case, the evaluation function is defined as follows.

$$E(X, W, V) = \frac{1}{2L} \sum_{l=1}^{L} \sum_{s=1}^{R} \left(d_s(\boldsymbol{x}^{(l)}) - h_s(\boldsymbol{x}^{(l)})\right)^2, \tag{4}$$

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 in Eq.(4). Using the Eqs. (5) and (6), $w_{ij} \in W$ and $v_{si} \in V$ are updated based on the BP. The flowchart of BP is shown as in Fig. 3 [42], where X, T_{max}, θ, and α denote the set of learning data, the maximum number of learning times, thresholds, and learning rates, respectively.

$$\begin{aligned} \Delta w_{ij} &= \alpha \sum_{s=1}^{R} (d_s(\boldsymbol{x}^{(l)}) - h_s(\boldsymbol{x}^{(l)}))(1 - h_s(\boldsymbol{x}^{(l)}))v_{si} \\ &\quad \times y_i(\boldsymbol{x}^{(l)})(1 - y_i(\boldsymbol{x}^{(l)}))x_j^{(l)} \quad (5) \\ \Delta v_{si} &= \alpha(d_s(\boldsymbol{x}^{(l)}) - h_s(\boldsymbol{x}^{(l)}))h_s(\boldsymbol{x}^{(l)}) \\ &\quad \times (1 - h_s(\boldsymbol{x}^{(l)}))y_i(\boldsymbol{x}^{(l)}) \quad (6) \end{aligned}$$

E. NG and k-means methods

In this section, we explain the NG method, which is an unsupervised learning method based on the SDM. Vector quantization approximates a large amount of data with a small amount of data by extracting features. There are many known

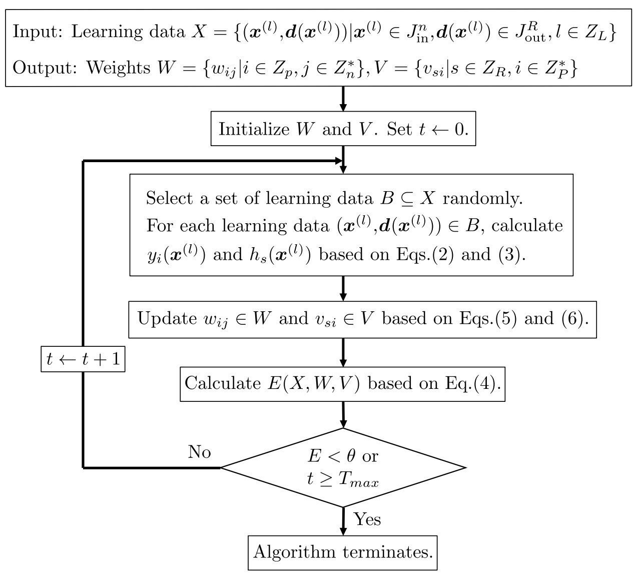
Input: Learning data $X = \{(\boldsymbol{x}^{(l)}, \boldsymbol{d}(\boldsymbol{x}^{(l)})) | \boldsymbol{x}^{(l)} \in J_{\text{in}}^n, \boldsymbol{d}(\boldsymbol{x}^{(l)}) \in J_{\text{out}}^R, l \in Z_L\}$
Output: Weights $W = \{w_{ij} | i \in Z_p, j \in Z_n^*\}, V = \{v_{si} | s \in Z_R, i \in Z_P^*\}$
Initialize W and V. Set $t \leftarrow 0$.
Select a set of learning data $B \subseteq X$ randomly.
For each learning data $(\boldsymbol{x}^{(l)}, \boldsymbol{d}(\boldsymbol{x}^{(l)})) \in B$, calculate $y_i(\boldsymbol{x}^{(l)})$ and $h_s(\boldsymbol{x}^{(l)})$ based on Eqs.(2) and (3).
Update $w_{ij} \in W$ and $v_{si} \in V$ based on Eqs.(5) and (6).
$t \leftarrow t+1$
Calculate $E(X, W, V)$ based on Eq.(4).
No
$E < \theta$ or $t \geq T_{max}$
Yes
Algorithm terminates.

Fig. 3. Algorithm of BP

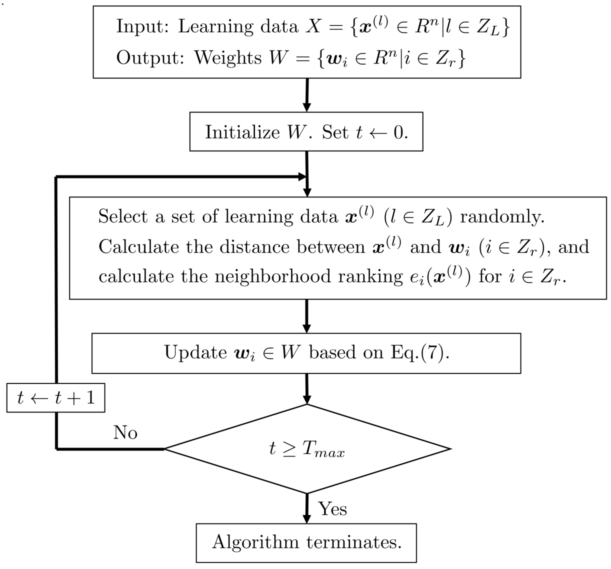
Input: Learning data $X = \{\boldsymbol{x}^{(l)} \in R^n | l \in Z_L\}$
Output: Weights $W = \{\boldsymbol{w}_i \in R^n | i \in Z_r\}$
Initialize W. Set $t \leftarrow 0$.
Select a set of learning data $\boldsymbol{x}^{(l)}$ $(l \in Z_L)$ randomly.
Calculate the distance between $\boldsymbol{x}^{(l)}$ and $\boldsymbol{w}_i$ $(i \in Z_r)$, and calculate the neighborhood ranking $e_i(\boldsymbol{x}^{(l)})$ for $i \in Z_r$.
Update $\boldsymbol{w}_i \in W$ based on Eq.(7).
$t \leftarrow t+1$
No
$t \geq T_{max}$
Yes
Algorithm terminates.

Fig. 4. Algorithm of NG

methods for vector quantization, of which NG and k-means are the most representative. Below, we explain the NG and its special case, the k-means method [43].

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 (also called cluster centers), where $|X| = L$, d and r are positive integers.

Let $e_i(\boldsymbol{x}) \in Z_{r-1}^*$ be 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 $\Delta \boldsymbol{w}_i$.

$$\Delta \boldsymbol{w}_i = \varepsilon \cdot \exp(-e_i(\boldsymbol{x})/\lambda) \cdot (\boldsymbol{x} - \boldsymbol{w}_i), \tag{7}$$

where $\varepsilon \in [0, 1]$ and λ is a positive real number. Eq.(7) indicates that the closer an element of W is to the input $\boldsymbol{x}$, the closer it is to $\boldsymbol{x}$. The set W approximates X by minimizing E.

For NG, this approximation is achieved by solving the minimization problem of the evaluation function E, as shown below.

$$E = \sum_{\boldsymbol{x}^{(l)} \in X} \sum_{i=1}^{r} \frac{\exp(-e_i(\boldsymbol{x})/\lambda)}{\sum_{i'=1}^{r} \exp(-e_{i'}(\boldsymbol{x})/\lambda)} ||\boldsymbol{x}^{(l)} - \boldsymbol{w}_i||^2 \tag{8}$$

If $\lambda \to 0$, 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 $\boldsymbol{x}$. For a given $\boldsymbol{x}$, the set W approximates X by repeatedly updating the elements of W that are closest to it.

The NG method is shown as in Fig. 4 [43], where T_{max} denotes the maximum number of learning times.

F. Federated Learning

Let us explain FL as a conventional learning method [15]–[17]. It is assumed that the system consists of $Q+1$ servers as in Fig. 1 and Server 0 is the central server. Initially, the data set X is partitioned into Q subsets and stored in each server, where $|X| = L$. Let B_q be a subset for Server q, where $X = \bigcup_{q=1}^{Q} B_q$. Server q uses its own dataset B_q to calculate the gradient coefficient of the parameter for this dataset. The number of elements in each dataset is $|B_q| = n_q$. The gradient coefficients of each server are sent to Server 0, where they are integrated, and the learning parameters for the data set X are updated. This result is sent to each server, and the gradient coefficients are updated using this result. In the same way, integrating the results and updating the parameters of Server 0 and the gradient coefficients of each server are repeated. Let C and Q be the rate for server selection and the number of servers, respectively. Then, a fundamental FL algorithm FedSGD described in [17] is shown below.

[**A**lgorithm of FL] FedSGD(X, P)
Input : The set X of learning data
Output : The set P of parameters
[**S**tep 1]
Select $r = \lceil C \times Q \rceil$ servers from Q servers. Let $k_j \in Z_Q$ $(j = 1, \cdots, r)$ be the number of j-th selected server.
[**S**tep 2]
In each server k_j, calculate $g_{k_j} = F(p, B_{k_j}) (p \in P)$, and send it to Server 0.
[**S**tep 3]
In Server 0, update the parameter $p \in P$ as Eq.(9) and send it to each server.

$$p \leftarrow p - \alpha \sum_{j=1}^{r} \frac{n_{k_j}}{L} g_{k_j} \tag{9}$$

[**S**tep 4]
Update the parameter $p \in P$ in each server.
[**S**tep 5]
If the learning completion condition is satisfied, the algorithm terminates. Otherwise, go to Step 1. □

If $C = 1$, the model is called a synchronous model; otherwise, if $C \neq 1$, the model is called an asynchronous model.

Table III shows the algorithm of FedSGD. On the method, each server stores B_q and does not store other data. In Table III, Q sets of learning data are not integrated. Therefore, the proposed method is more secure than the case where all the data is held in a single server.

III. Secure learning method by using divided data

In this section, we propose a learning method for distributed processing using simple, secure, divided data as a fourth method to preserve privacy. The features of the proposed method are as follows.

1) The learning data are randomly divided and placed on each server to realize learning in a distributed environment. The original data is never used during the training. In other words, a large amount of data owned by users can be used for machine learning while they are securely stored in the servers.
2) Divided parameters are securely stored in the servers. They are updated on each server but not exchanged throughout the learning process.
3) The results (parameters) can be retrieved at any point in time.

In Section III-A, we show that such a method of updating parameters is feasible for machine learning. That is, we show that the integration of partial updates of divided parameters by divided data progressively reduces the overall error (objective function). In Sections III-B and C, we propose BP learning and NG (k-means) methods based on this learning. The former, an example of supervised learning, estimates parameters that approximate input-output relations while maintaining the confidentiality of the learning data. The latter is an example of unsupervised learning in which a small number of parameters approximates the distribution of the given learning data. In Section III-D, we compare the conventional and proposed methods and clarify their characteristics. Throughout this section, we show why the proposed method can achieve a very high level of confidentiality. The reason is that the proposed method stores data on servers in a segmented state whereas conventional methods, including FL, store data on servers as they are.

A. SDM using divided data and parameters

In this section, the proposed method of parameter updating is theoretically proven to be feasible for machine learning. In the proposed method, the learning data and parameters are divided in advance and stored in each server. The proposed method performs learning by repeating the process of integrating the results of the parameters updated by each server at the central server. Then, we consider whether the steepest descent method that updates and integrates divided parameters work wells overall. In other words, does the overall error progressively decrease as learning proceeds? As described in Section II-C, the parameter update equation of the steepest descent method is generally given as follows [41].

$$\frac{d\theta_i}{dt} = -\frac{\partial E}{\partial \theta_i} \text{ for } i = 1, \cdots, m, \tag{10}$$

where E is the objective function and m is the number of parameters.

If the parameter is divided into Q pieces,

$$\theta_i = g(\theta_{i1}, \cdots, \theta_{iQ}) \text{ for } i = 1, \cdots, m. \tag{11}$$

That is, θ_i is defined as the function of divided parameters. The updated equation of the divided parameters for this model is assumed as follows.

$$\frac{d\theta_{iq}}{dt} = -\frac{\partial E}{\partial \theta_{iq}} \text{ for } i = 1, \cdots, n, \ q = 1, \cdots, Q \tag{12}$$

In this case, the following relation holds.

$$\frac{dE}{dt} \leq 0, \tag{13}$$

where equality holds if the right-hand side of Eq.(12) is zero. Because the left-hand side of Eq.(13) can be rewritten as follows.

$$\frac{dE}{dt} = \sum_{i,q} \frac{\partial E}{\partial \theta_i} \frac{\partial \theta_i}{\partial \theta_{iq}} \frac{d\theta_{iq}}{dt}. \tag{14}$$

Then, we rewrite Eq.(12) as follows.

$$\frac{d\theta_{iq}}{dt} = -\frac{\partial E}{\partial \theta_i} \frac{\partial \theta_i}{\partial \theta_{iq}} \text{ for } q = 1, \cdots, Q. \tag{15}$$

By using it, we rewrite Eq.(14) as follows:.

$$\frac{dE}{dt} = -\sum_{i,q} \left(\frac{\partial E}{\partial \theta_i} \times \frac{\partial \theta_i}{\partial \theta_{iq}} \right)^2 \leq 0. \tag{16}$$

This result shows that the update of the divided parameters on each server (Eq.(12)) leads to a reduction in the overall error (Eq.(16)). Based on this result, we propose to update the divided parameters instead of the original non-divided parameters.

B. Distributed system with secure divided data and parameters for BP

It is assumed that the system consists of $Q+1$ servers as in Fig. 1 and Server 0 is the central server. In this section, any learning data $(\boldsymbol{x}^{(l)}, \boldsymbol{d}(\boldsymbol{x}^{(l)})) \in X$ is divided into Q pieces and they are stored in each server as follows [39], [40].

$$x_j^{(l)} = \Pi_{q=1}^{Q} x_{jq}^{(l)}, \tag{17}$$

$$d_s(\boldsymbol{x}^{(l)}) = \sum_{q=1}^{Q} d_{sq}(\boldsymbol{x}^{(l)}). \tag{18}$$

Each weight of the sets W and V is divided in the same manner into Q pieces and stored in each server as follows.

$$w_{ij} = \Pi_{q=1}^{Q} w_{ijq}, \tag{19}$$

$$v_{si} = \Pi_{q=1}^{Q} v_{siq}. \tag{20}$$

Let $W_q = \{w_{ijq} | i \in Z_P, j \in Z_n^*\}$ and $V_q = \{v_{siq} | s \in Z_S, i \in Z_P^*\}$. Let $\Pi_{q=1}^{Q} x_{0q} = 1$.

TABLE III
ALGORITHM OF FEDSGD

	Server 0	Server q $(q\in Z_Q)$
Initialize	Initialize $\{w_{ij}, v_{si}\}$ and send the results to each server.	Store $B_q\subseteq X$. Store $\{w_{ij}, v_{si}\}$.
Step 1	Set $t\leftarrow 0$	
Step 2	Select $U\subseteq Z_Q$ and send the results to each server.	
Step 3		If $q\in U$, based on Eqs.(2) and (3), calculate $\Delta_{1ijq}=\alpha\sum_{\boldsymbol{x}^{(l)}\in B_q}\sum_{s=1}^{R}(d_s(\boldsymbol{x}^{(l)})-h_s(\boldsymbol{x}^{(l)}))(1-h_s(\boldsymbol{x}^{(l)}))v_{si}$ $\times y_i(\boldsymbol{x}^{(l)})(1-y_i(\boldsymbol{x}^{(l)}))x_j^{(l)}$, $\Delta_{2siq}=\alpha\sum_{\boldsymbol{x}^{(l)}\in B_q}(d_s(\boldsymbol{x}^{(l)})-h_s(\boldsymbol{x}^{(l)}))h_s(\boldsymbol{x}^{(l)})(1-h_s(\boldsymbol{x}^{(l)}))y_i(\boldsymbol{x}^{(l)})$ and send the results to Server 0.
Step 4	Calculate $\Delta_{3ij}=\sum_{q\in U}\Delta_{1ijq}\times\lvert B_q\rvert/\lvert X\rvert$ and $\Delta_{4si}=\sum_{q\in U}\Delta_{2siq}\times\lvert B_q\rvert/\lvert X\rvert$ and send to each server.	
Step 5		Update $\{w_{ij}, v_{si}\}$ as follows : $w_{ij}\leftarrow w_{ij}+\Delta_{3ij}$ $v_{si}\leftarrow v_{si}+\Delta_{4si}$ Calculate E_q as follows and send the results to Server 0. $E_q=\sum_{\boldsymbol{x}\in B_q}\sum_{s=1}^{R}\left(d_s(\boldsymbol{x}^{(l)})-h_s(\boldsymbol{x}^{(l)})\right)^2$
Step 6	Calculate $E=\frac{1}{L}\sum_{q=1}^{Q}E_q$. If $E<\theta$ or $t=T_{max}$, the algorithm terminates. Set $t\leftarrow t+1$ and go to Step 2.	

An output of the NN for the divided data, based on Eqs.(17), (18), (19), and (20) is calculated instead of Eqs.(2) and (3) as follows.

$$y_i(\boldsymbol{x})=\frac{1}{1+\exp\left(-\left(\sum_{j=0}^{n}\Pi_{q=1}^{Q}w_{ijq}x_{jq}\right)\right)}. \quad (21)$$

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

$$h_s(\boldsymbol{x})=\frac{1}{1+\exp\left(-\left(\sum_{i=0}^{P}\Pi_{q=1}^{Q}v_{siq}y_{iq}\right)\right)}. \quad (22)$$

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

In this case, the MSE for the set of learning data is calculated as follows.

$$E(X)=\frac{1}{2L}\sum_{l=1}^{L}\sum_{s=1}^{R}\left(\sum_{q=1}^{Q}(d_{sq}(\boldsymbol{x}^{(l)})-h_{sq}(\boldsymbol{x}^{(l)}))\right)^2. \quad (23)$$

Based on BP, each of the weights w_{ijq} $(i\in Z_P, j\in Z_P^*)$ and v_{siq} $(s\in Z_R, i\in Z_P^*)$ is updated instead of Eqs.(5) and (6) as follows [40].

$$\begin{aligned}\Delta w_{ijq} &= \alpha\sum_{\boldsymbol{x}^{(l)}\in B}\sum_{s=1}^{R}\left(\sum_{q=1}^{Q}(d_{sq}(\boldsymbol{x}^{(l)})-h_{sq}(\boldsymbol{x}^{(l)}))\right)\\ &\times (1-h_s(\boldsymbol{x}^{(l)}))\Pi_{q=1}^{Q}v_{siq}y_{iq}(\boldsymbol{x}^{(l)})(1-y_i(\boldsymbol{x}^{(l)}))\\ &\times (\Pi_{q=1}^{Q}w_{ijq}x_{jq}^{(l)})/w_{ijq}, \quad (24)\end{aligned}$$

$$\begin{aligned}v_{siq} &= \alpha\sum_{\boldsymbol{x}^{(l)}\in B}\sum_{q=1}^{Q}(d_{sq}(\boldsymbol{x}^{(l)})-h_{sq}(\boldsymbol{x}^{(l)}))h_s(\boldsymbol{x}^{(l)})\\ &\times (1-h_s(\boldsymbol{x}^{(l)}))(\Pi_{q=1}^{Q}y_{iq}v_{siq})/v_{siq}. \quad (25)\end{aligned}$$

Note that, unlike the conventional method, the right-hand side of Eqs. (24) and (25) involve factors of $1/w_{ijq}$ and $1/v_{siq}$.

The outline of the learning method based on the divided data shown here is as follows. Note that each parameter $p\in P$ is assumed to be divided into $p=\Pi_{q=1}^{Q}p_q$.

[The outline of the proposed BP method]
Input : The set X of learning data
Output : The set P of parameters
[Step 1]
All data and parameters are divided into multiple pieces, and each is sent to Server $q\in Z_Q$. Let X_q be the set of data sent to Server q and P_q be the set of parameters of Server q.
[Step 2]
Select the set of data $B\subseteq X$ to be used to update the parameters. Let $B_1,\cdots,B_Q$ be the set of data obtained from B.
[Step 3]
In Server q for $q\in Z_Q$, compute $g_q=F(p_q,B_q)$ for $p_q\in Pq$ and sends it to Server 0.
[Step 4]
In Server 0, g_q is integrated, and the update amount Δp is calculated and sent to Server q for $q\in Z_Q$.
[Step 5]
In Server q, update $p_q\in P_q$ as follows.

$$p_q\leftarrow p_q+\Delta p/p_q \quad (26)$$

[Step 6]
If the learning completion is satisfied, the algorithm terminates. Otherwise, go to Step 2. □

Table IV shows the detailed algorithm of the proposed BP method. The algorithm is online learning if $\lvert B\rvert=1$, batch learning if $\lvert B\rvert=\lvert X\rvert$, and mini-batch learning if $\lvert B\rvert<\lvert X\rvert$. All data and parameters are randomly divided and sent to each server. The algorithm in Table IV is as follows. In Step 2, each server calculates the product of the weights and inputs. In Step 3, Server 0 calculates the outputs of the intermediate layer by integrating the products of Step 2, divides the calculated output, and sends them to each server. In Step 4, each server calculates the product of the weights and outputs of the intermediate layer. In Step 5, Server 0 calculates the outputs of the output layer by integrating the results of Step 4, divides randomly each output, and sends them to each server. In Step 6, each server calculates the differences between the learning data and the output layer's outputs and sends them to Server 0. In Step 7, Server 0 calculates the error of NN by using the differences obtained in Step 6 and judges whether learning has been completed. If not, Server 0 determines the subset U of data numbers used in the following learning process. In addition, Server 0 calculates the error for the learning data corresponding to subset U, calculates the update amounts, and sends them to each server. Finally, in Step 8, each server updates its own parameter. Note that the constant a in Step 6 is used to keep the output data secret from Server 0.

In Table IV, the number of computations and the number of communications between servers are kept low by updating the weights when computing MSE (Step 7).

C. Distributed system with secure divided data and parameters for NG

It is assumed that the system consists of $Q+1$ servers as in Fig. 1 and Server 0 is the central server. We propose the NG algorithm as an unsupervised learning method using simple, secure, divided data. The k-means method can be realized as a special case of NG. To realize NG, we must 1) determine the neighborhood rank of each element $\boldsymbol{w}_i$ of the set W of the reference vectors with respect to the input data $\boldsymbol{x}^{(l)}$, and 2) update all elements $\boldsymbol{w}_i$ of the set W. The two steps of updating $\boldsymbol{w}_i$ are realized 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 $\boldsymbol{x}^{(l)} = (x_1^{(l)}, \cdots, x_j^{(l)}, \cdots, x_n^{(l)})$, $x_j^{(l)} = \sum_{q=1}^{Q} x_{jq}^{(l)}$, and $\boldsymbol{w}_i = (w_{i1}, \cdots, w_{ij}, \cdots, w_{in})$, $w_{ij} = \sum_{q=1}^{Q} w_{ijq}$.

[The outline of the proposed NG method]
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
[Step 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, calculate the distance D between input data and reference vector in each server and integrate in Server 0 as follows.

$$D_{ijq}^{(l)} = \left(x_{jq}^{(l)} - w_{ijq} \right) \tag{27}$$
$$(j = 1, \cdots, n, q = 1, \cdots, Q).$$

In Server 0, calculate as follows.

$$||\boldsymbol{x}^{(l)} - \boldsymbol{w}_i||^2 = \sum_{j=1}^{n} \left(\sum_{q=1}^{Q} D_{ijq}^{(l)} \right)^2 \tag{28}$$

[Step 2]

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

$$\begin{aligned} \Delta w_{ijq} &= \frac{\partial E}{\partial w_{ijq}} = \frac{\partial E}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial w_{ijq}} \\ &= \varepsilon \cdot \exp(-e_i(\boldsymbol{x}^{(l)})/\lambda) \cdot (x_j^{(l)} - w_{ij}) \end{aligned} \tag{29}$$

[Step 3]

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

Table V shows the detailed algorithm of the proposed NG. 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 U of the natural numbers is selected randomly. In Step 2, we calculate $D_{ijq}^{(l)}$ in each server and send it to Server 0. In Step 3, we calculate the update amount Δw_{ijq} of reference vector w_{ijq} by integrating $D_{ijq}^{(l)}$ 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_{max}, the algorithm terminates.

D. Differences between the proposed and conventional methods

Conventional machine learning methods with privacy-preserving or secret data including 1) SMC, 2) quasi-homomorphic encryption, and 3) federated learning [10], [13], [15] are compared with the proposed method.

In Method 1), each data sample is first divided into several pieces using a random number, and each piece is stored in each server. Then, the cooperative calculation is repeated among the servers using each piece of data held by each server. In this case, the data is kept divided, and the cooperative computation among servers performs learning by using only the operations that preserve confidentiality so that privacy is strictly preserved. In addition, this method can be applied to all the problems considered, in principle. However, as the scale of the problem and the amount of data increases, the method requires more computation time, limiting its utilization to various problems [22].

In Method 2), the data are first encrypted. Next, the necessary calculations for machine learning were performed using the encrypted data, and the final results were decrypted to perform the calculations. In this method, our goal is to find an encryption method that enables us to perform the computation when the data are encrypted. Similar to Method 1), it strictly protects privacy, but its utilization in various machine learning problems is limited [26].

In Method 3), the dataset is first partitioned into several subsets, each of which is stored in a server. Then, each server performs the necessary machine learning computations independently using these subsets and sends the results to the central server. In the following, the computation process is repeated to obtain partial results at each server and the overall result at the central server. In this method, the overall

TABLE IV
ALGORITHM OF THE PROPOSED BP METHOD (ONLINE, BATCH, AND MINI-BATCH LEARNING)

	Server 0	Server q ($q{\in}Z_Q$)
Initialize		Store $\{x_{jq}^{(l)}\|l{\in}Z_L, j{\in}Z_n\}$, $\{d_{sq}(\boldsymbol{x}^{(l)})\|l{\in}Z_L, s{\in}Z_R\}$. Initialize $\{w_{ijq}\|i{\in}Z_P, j{\in}Z_n^*\}$, $\{v_{siq}\|s{\in}Z_R, i{\in}Z_P^*\}$.
Step 1	$t{\leftarrow}0$	
Step 2		Calculate $w_{ijq}x_{jq}^{(l)}$ $(l{\in}Z_L, i{\in}Z_P, j{\in}Z_n^*)$ and send to Server 0.
Step 3	Calculate $y_i(\boldsymbol{x}^{(l)})$ as Eq.(21) and divide as $y_i = \Pi_{q=1}^{Q} y_{iq}$. Send $y_{iq}(q{\in}Z_Q)$ to each server.	
Step 4		Calculate $v_{siq}y_{iq}$ $(s{\in}Z_R, i{\in}Z_P^*)$ and send to Server 0.
Step 5	Calculate $h_s(\boldsymbol{x}^{(l)})$ $(s{\in}Z_R)$ as Eq.(22). Divide $h_s(\boldsymbol{x}^{(l)}) = \Pi_{q=1}^{Q} h_{sq}(\boldsymbol{x}^{(l)})$ and send $h_{sq}(\boldsymbol{x}^{(l)})$ $(q{\in}Z_Q)$ to each server.	
Step 6		Calculate $a(d_{sq}(\boldsymbol{x}^{(l)}) - h_{sq}(\boldsymbol{x}^{(l)}))$ $(s{\in}Z_R)$ and send to Server 0.
Step 7	Calculate $E(X, W, V)$ using Eq.(23). If $E < \theta$ or $t = T_{max}$, the algorithm terminates. Otherwise, select $U{\subseteq}Z_L$ randomly and calculate p_{1ij} and p_{2si} as follows : $p_{1ij} = \sum_{l\in U}\sum_{s=1}^{R}\sum_{q=1}^{Q} a(d_{sq}(\boldsymbol{x}^{(l)}) - h_{sq}(\boldsymbol{x}^{(l)}))h_{sq}(\boldsymbol{x}^{(l)})$ $\times(1 - h_s(\boldsymbol{x}^{(l)}))v_{si}y_i(\boldsymbol{x}^{(l)})(1 - y_i(\boldsymbol{x}^{(l)}))(\Pi_{q=1}^{Q} w_{ijq}x_{jq}^{(l)})$, $p_{2si} = \sum_{l\in U}\sum_{q=1}^{Q} a(d_{sq}(\boldsymbol{x}^{(l)}) - h_{sq}(\boldsymbol{x}^{(l)}))h_s(\boldsymbol{x}^{(l)})$ $\times(1 - h_s(\boldsymbol{x}^{(l)}))(\Pi_{q=1}^{Q} y_{iq}v_{siq})$ p_{1ij} and p_{2si} are sent to each server.	
Step 8		Using p_{1ij}, p_{2si}, update $\{w_{ijq}\|i{\in}Z_P, j{\in}Z_j^*\}$ and $\{v_{siq}\|s{\in}Z_R, i{\in}Z_P^*\}$ as follows. $w_{ijq}{\leftarrow}w_{ijq} + \alpha p_{1ij}/a w_{ijq}$ $v_{siq}{\leftarrow}v_{siq} + \alpha p_{2(i)}/a v_{siq}$ Set $t{\leftarrow}t + 1$ and go to Step 2.

TABLE V
ALGORITHM OF THE PROPOSED NG METHOD.

	Server 0	Server q
Initialize	Determine the values ε_{int}, ε_{fin}. Set $t = 1$.	Store $\{x_{jq}^{(l)}\|l{\in}Z_L, j \in Z_n\}$. Initialize $\{w_{ijq}\|i{\in}Z_r, j \in Z_n\}$.
Step 1	Select the set of natural numbers $U{\subseteq}Z_L$ randomly and send to each server.	
Step 2		Calculate $D_{ijq}^{(l)} = (x_{jq}^{(l)} - w_{ijq})$ $(l{\in}U, i{\in}Z_r, j{\in}Z_n)$ and send to Server 0.
Step 3	Based on Eq.(28), calculate $\exp(-e_i(\boldsymbol{x}^{(l)})/\lambda)$ and $\Delta w_{ij} = \varepsilon_{int}\left(\frac{\varepsilon_{fin}}{\varepsilon_{int}}\right)^{\frac{t}{T}}\sum_{l\in U}\exp(-e_i(\boldsymbol{x}^{(l)})/\lambda)\sum_{q=1}^{Q} D_{ijq}^{(l)}$ and send to each server.	
Step 4		Update $\{w_{ijq}\|i{\in}Z_r, j{\in}Z_n\}$ as follows. $w_{ijq}{\leftarrow}w_{ijq} + \Delta w_{ij}$
Step 5	If $t = T_{max}$, the algorithm terminates. Otherwise, Set $t{\leftarrow}t + 1$ and go to Step 2.	

problem is decomposed into several smaller problems, and the overall solution is repeatedly estimated using the results of each problem. This method can respond flexibly to the scale of problems and data by increasing the number of partitions. Security issues due to the exchange of data between servers have been pointed out [27]. Furthermore, with respect to these studies, many studies have improved the usability of SMC, and many models have been proposed to improve the problem of FL [19]–[21].

In the proposed method, each of data and parameters is divided into several pieces using a random number and stored in each server. Learning is achieved by repeatedly executing independent partial computations on each server and the exact integrated computations on the central server. Therefore, this method is highly useful for many machine learning problems, such as FL. In addition, this method is more secure than FL because it learns from the divided data and parameters instead of the entire dataset itself in each server, as in FL. In our model, the user is on Server 0, and the divided data and parameters are deposited on other servers in a distributed manner. The malicious adversary is assumed to reside on the server where the data is deposited. Since the data and parameters remain divided, confidentiality is maintained against the adversary on the server hosting the data. Another possible security threat is related to integrity, such as data tampering. This issue is to be addressed in the future.

In summary, Methods 1) and 2) are designed to realize machine learning while strictly preserving confidentiality by using data encryption and random numbers, whereas Method 3) and the proposed method distribute the data to each server

TABLE VI
THE DATASET FOR PATTERN CLASSIFICATION

	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

and execute learning by distributed processing without taking the data itself out of the server. Therefore, Methods 1) and 2) are highly secure, but their usability in various machine learning problems is limited. Method 3) and the proposed method are efficiently utilized in many machine learning problems and are also highly adaptable to edge computing for IoT applications. However, the security level of the proposed method is lower than that of Methods 1) and 2). Therefore, Method 3) and the proposed method can be combined with Methods 1) and 2), and methods such as differential privacy to enhance confidentiality. Comparing the security level of the proposed method with that of Method 3), the proposed method, which does not directly use data to perform learning, is superior. From these results, we can say that the performance of the proposed method is in the middle of Methods 1), 2) and 3). Furthermore, from the viewpoint of safe and secure long-term storage of big data, Method 1) and the proposed method are superior to Method 3). Through the numerical experiments in Section IV, it is shown that the proposed method based on distributed processing using SDM with divided data has the same level of accuracy as the conventional learning method using all data as it is.

IV. NUMERICAL SIMULATIONS

In this section, we compare the learning accuracies of the proposed method with those of the conventional methods and the FL through numerical experiments on benchmark problems. In Table VI, #data, #input, and #class mean the numbers of data, input, and class for each data, respectively.

A. Simulation for the proposed BP

To demonstrate that the proposed BP can achieve sufficient accuracy compared to the conventional method, we classify the five benchmark datasets, Iris, Wine, Sonar, BCW, and Spam [44]. In this section, $P = 10$ and $R = 3$ for Iris and Wine, and $R = 2$ for Sonar, BCW, and Spam in Eqs.(2) and (3). In the proposed method, $Q = 3$ and $a = 1$. In this simulation, 5-fold cross-validation as the evaluation method is used. For each method, the maximum number of learning iterations is $T_{max} = 50000$, and the learning rates are $K_w = 0.01$ and $K_v = 0.01$. If the learning time is T_{max} or the mean square error (MSE) is less than the threshold, each algorithm terminates. The threshold is $\theta = 3.0\times10^{-2}$ for Iris and Wine, $\theta = 4.0\times10^{-2}$ for Sonar and BCW, and $\theta = 1.0\times10^{-1}$ for Spam, respectively.

After learning, we compared the conventional and proposed BP methods in terms of the misclassification rate and the number of learning times.

TABLE VII
RESULTS OF SIMULATIONS USING CONVENTIONAL AND PROPOSED BP METHODS

		Iris	Wine	Sonar	BCW	Spam
A1	Learn(%)	2.41	0.51	1.81	2.38	6.34
	Test(%)	3.77	1.75	16.71	2.93	7.02
	LT	48799	24468	49211	25448	50000
A2	Learn(%)	1.73	0.50	1.19	2.34	6.18
	Test(%)	3.23	2.28	17.55	3.01	7.24
	LT	15371	1681	5150.08	920	4007
A3	Learn(%)	1.71	0.53	1.28	2.31	5.86
	Test(%)	2.83	2.06	15.74	2.91	6.70
	LT	28108	5004	12722	2884	2569
B1	Learn(%)	2.45	0.53	1.77	2.37	8.19
	Test(%)	3.20	2.08	16.43	2.89	8.59
	LT	48462	24411	49285	26006	50000
B2	Learn(%)	1.66	0.54	1.19	2.30	6.18
	Test(%)	3.00	1.92	16.05	2.91	7.29
	LT	15474	1682	4972	1313	3986
B3	Learn(%)	1.68	0.51	1.17	2.31	5.81
	Test(%)	3.07	2.14	16.52	2.86	6.64
	LT	27798	5116	12664	3413	4585
C1	Learn(%)	4.02	1.19	3.86	2.30	6.41
	Test(%)	4.87	3.58	18.83	2.99	7.19
	LT	49474	16333	43298	20489	50000
C2	Learn(%)	2.73	1.23	1.95	2.23	6.14
	Test(%)	5.33	3.97	18.14	3.01	6.91
	LT	6513	1057	7295	576	901
C3	Learn(%)	1.74	1.12	2.31	2.22	5.96
	Test(%)	4.03	3.97	18.38	3.02	6.71
	LT	13051	1919	20986	1621	1444

Table VII shows the results of the experiments on online learning, batch learning, and mini-batch learning. In the table, A1 is the normal online BP method, A2 is the normal batch method, A3 is the normal mini-batch BP method, B1 is the online BP method for HPD, B2 is the batch BP method for HPD, B3 is the mini-batch BP method for HPD, C1 is the online BP method using Table IV, C2 is the batch BP method using Table IV, and C3 is the mini-batch BP method using Table IV. In the mini-batch BP method, $1/3$ of the learning data are randomly selected to update the weights. Learn and Test are the misclassification rates (%) for the learning and the test data, respectively, and LT is the number of times the weights are updated at the end of learning. The values in Table VII are the averages of twenty trials each.

The results in Table VII show that the proposed method is almost as accurate as the conventional BP and BP methods combined with the conventional BP method.

B. Simulation for the proposed NG

In this section, we show that the proposed method can achieve sufficient accuracy compared to conventional methods. To demonstrate that the proposed method can achieve sufficient accuracy compared to the conventional method, we perform clustering the five benchmark datasets, Iris, Wine, Sonar, BCW, and Spam [44], using the conventional NG method and the proposed NG method.

Here, the number r of reference vectors is 3 in the case of Iris and Wine and 2 in the case of Sonar, BCW, and Spam. In the proposed method, we set $Q = 3$. The maximum number of learning was set to 15000 for Iris, 18000 for Wine, 21000 for Sonar, 70000 for BCW, and 50000 for Spam. In

TABLE VIII
CALCULATION RESULTS BY USING CONVENTIONAL AND PROPOSED NG METHODS

		Iris	Wine	Sonar	BCW	Spam
A1	GP(%)	4.1	7.0	45.0	3.7	26.8
	MSE	2.91	25.53	288.38	200.64	798.24
A2	GP(%)	4.0	6.9	45.3	3.5	26.0
	MSE	2.84	24.89	280.95	195.40	782.25
A3	GP(%)	4.0	7.3	45.0	3.5	25.1
	MSE	2.84	24.90	281.21	195.49	782.35
B1	GP(%)	4.0	7.0	45.2	3.6	20.8
	MSE	2.87	25.07	284.00	198.14	785.84
B2	GP(%)	4.0	6.6	45.1	3.5	25.1
	MSE	2.84	24.89	280.94	195.40	782.47
B3	GP(%)	4.0	7.3	44.7	3.5	25.1
	MSE	2.84	24.88	280.96	195.44	781.70

the experiments, we considered that learning was completed when the number of updates of the reference vector reached the maximum number of learning times.

After learning, we compared the conventional and proposed NG methods in terms of the global purity (GP) $\frac{1}{L}\sum_{i\in Z_r}\max_{j\in Z_r}(n_{i,j})\times 100$ (%) and the evaluation function of Eq.(8), where $n_{i,j}$ is the number of data belonging to the i-th cluster and the j-th actual class.

The results of our experiments on online, batch, and mini-batch learning are presented in Table VIII, where A1 is the conventional online NG method, A2 is the conventional batch NG method, A3 is the conventional mini-batch NG method, B1 is the online NG method in Table V, B2 is the batch NG method in Table V, and B3 is the mini-batch NG method in Table V. In the mini-batch BP method, $1/3$ of the learning data were randomly selected to update the weights. GP and MSE are the GP value (%) and the value of Eq.(8). The values in the table represent the average of 20 trials each.

In Table VIII, the GP and the evaluation function values (accuracy) of the NG method are equivalent to those of the conventional methods. Through the numerical experiments in Section IV, it is shown that the proposed method based on distributed processing using SDM with divided data has the same level of accuracy as the conventional learning method using all data as it is.

V. CONCLUSION

A. Contribution of this study to society

The goal of the SDGs is to build a sustainable society, and Japan will contribute to this goal through the realization of a super-smart society, aiming at Society 5.0. How, then, will safe and secure machine learning, including this research, contribute to super-smart sustainable societies? Water and air are essential elements for building a sustainable society, and we have been creating a safe and secure environment for hundreds of years. On the other hand, since the 20th century, with the development of computers and communications, individuals have come to live in direct relationships with the country and the world. In the past, the relationship between people was based on the exchange of information in person or by mail. Today, we live in an age where information can be transmitted and received instantly across national borders. In this era, personal information can be easily collected and converted into big data, and new knowledge and information from these data will be generated progressively through analysis by AI and returned to individuals and society. In addition, big data will be safely stored for subsequent reuse to support a sustainable society. This is the desired form of the so-called super-smart society.

What is necessary for the sustainability of the big data processing cycle in this society? For this purpose, it is necessary to create a system environment in which this cycle can be utilized safely and securely. Many studies on machine learning for privacy preservation, including the present work, will contribute to the construction of infrastructure in this field, and users will be freed from anxiety regarding personal identification and information leakage in the cycle of providing individual information and obtaining knowledge and information.

For example, human flow data, which have recently generated considerable discussion, allow us to understand when, where, and how many people are present. Big data can be easily collected using location information from smartphones and camera images. This information is generally expected to be used for marketing, tourism, administrative services, and disaster prevention. For COVID-19, it is possible to visualize the data distribution of infected people, clusters, and the spread of infected people by applying statistical and AI processing to these data. Through these results, we can obtain knowledge and guidelines on how to act safely and securely. However, because this information is highly private, it must be handled carefully by quantifying it so that individuals cannot be identified. The following is a summary of how the conventional and proposed methods provide safety and security to users.

1) Although the encryption method is a safe and secure approach that does not strictly allow users to be identified, it is considered to increase the difficulty of advanced AI processing and the delay in response due to the increase in big data. In other words, it is challenging to process big data sent from ever-changing areas (disaster areas) in real time and accurately. Therefore, many studies have been conducted on improving the usability of data while maintaining the security of this method.

2) Although FL can flexibly respond to advanced AI processing associated with the increase in big data, the problem of information leakage and dissipation in each server where big data is distributed is a concern, as in conventional cloud systems. Therefore, many studies are being conducted on how to combine security assurance methods such as encryption when exchanging data between servers.

3) The processing power of the proposed method is almost the same as that of FL for increasing amounts of big data. In addition, personal data are divided into pieces in advance, and machine learning using these pieces can reduce the possibility of leakage and dissipation of personal information. In other words, in a super-smart society using the proposed method, user information is safely and securely incorporated into social computational systems in the form of division and storage and is securely processed by advanced AI. Users can obtain information on their surroundings and safe behavior patterns

of individuals, as needed. In addition, the provided information is safely and securely stored in a form that does not identify the individual for subsequent reuse.

Similarly, the application of privacy-preserving AI processing is expanding in various fields such as medicine, disaster prevention and mitigation, education, industry, agriculture, and the financial sector. Therefore, the future of advanced AI processing of big data requires a common understanding of the balance between the value of utilizing AI and the risk of privacy in a super-smart society. In addition, further studies from various perspectives are desirable to support the emergence of an era in which the sustainable cycle in these fields is provided as a stable and continuous infrastructure. Therefore, many studies on machine learning for privacy preservation, including this study, will contribute to the construction of infrastructure in this field.

B. Status and future development of the proposed method

In this paper, we have proposed a method of machine learning with data confidentiality and demonstrate its effectiveness by applying it to the BP method (supervised learning) and the NG method (unsupervised learning). In other words, we have shown that the SDM-based learning methods for distributed processing using divided data and parameters can achieve the same level of accuracy as the conventional learning methods. In AI processing and visualization of big data, there is a mixture of expectations of results and anxiety about providing data. To remove this anxiety, two conventional methods are known; the first is to encrypt the data and perform machine learning to protect privacy strictly, and the other is to partition the data into multiple subsets and integrate the results of each learning, which facilitates learning. In the proposed method, each data sample is divided into several random pieces in advance, and each piece is stored in multiple servers. Learning is realized by distributed processing between the central server and multiple servers.

The advantage of the proposed method is that it reduces the risk of privacy violation by dividing the original data during learning, as in the encryption method, and it is utilizable in many different problems, as in the case of FL. We summarize our conclusions below.
1) The need for advanced AI processing of big data to support the realization of a sustainable society is increasing in many fields. So far, many studies have focused on the value of utilization of AI processing, while others have focused on the associated security risk.
2) The approach proposed in this study is a learning method that strikes a good balance between the value for utilization and the security risk of AI processing.

In the future, we would like to develop a learning method with high processing capability in terms of both utilization and risk, aiming to build a safe and secure infrastructure for advanced AI processing in a super-smart society. Technically, we would like to improve the security level of the proposed method by introducing SMC, encryption, and differential privacy, and to develop a new model that combines the proposed method with the FL.

Acknowledgment

This work was partially supported by a research grant from Telecommunications Advancement Foundation (TAF) and Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 20K11994.

References

[1] United Nations Foundation, *SUSTAINABLE DEVELOPMENT GOALS*, https://unfoundation.org/, 2021.
[2] UNDP Ukraine, *Transforming our world: the 2030 Agenda for Sustainable Development*, https://www.ua.undp.org/, 2021
[3] Cabinet Office of Japan, *Society 5.0*, https://www.cao.go.jp/, 2021.
[4] United Nations, *Global Sustainable Development Report*, 2015 edition, https://www.un.org/en/desa, 2021
[5] C. C. Aggarwal and P. S. Yu, *Privacy-Preserving Data Mining: Models and Algorithms*, ISBN 978-0-387-70991-8, Springer-Verlag, 2009.
[6] A. Shamir, *How to share a secret*, Comm. ACM, vol. 22, no. 11, pp. 612-613, 1979.
[7] A. Beimel, *Secret-sharing schemes: a survey*, in Proc. of the Third international conference on Coding and cryptology (IWCC 11), 2011.
[8] W. Stallings, *Cryptography and Network Security Principles and Practices (7th ed.)*, Pearson Education, Inc., 2017.
[9] 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, 2022.
[10] R. Canetti, U. Feige, O. Goldreich, and M. Naor, *Adaptively secure multi-party computation*, STOC' 96, pp. 639-648, 1996.
[11] R. Cramer, I. Damgård, and U. Maurer, *General secure multi-party computation from any linear secret-sharing scheme*, EUROCRYPT', pp.331-339, 2000.
[12] A. Ben-David, N. Nisan, and B. Pinkas, *Fair play MP: a system for secure multi-party computation*, ACM CCS' 08, pp. 257-266, 2008.
[13] C. Gentry, *Fully Homomorphic Encryption Using Ideal Lattices, STOC2009*, pp.169-178, 2009.
[14] HElib, *An Implementation of homomorphic encryption*, https://github.com/shaih/HElib
[15] 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.
[16] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, *Federated Learning: Strategies for Improving Communication Efficiency*, arXiv:1610.05492, 2017.
[17] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B.A.y. Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*, Proc. of Machine Learning Research, vol. 54, pp. 1273-1282, 2017.
[18] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, *Differentially private empirical risk minimization*, Journal of Machine Learning Research, vol. 12, no. 29, pp. 1069-1109, 2011.
[19] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, *Partially Encrypted Multi-Party Computation for Federated Learning*, IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), DOI: 10.1109/CCGrid51090.2021.00101, 2021.
[20] J. Duan, J. Zhou, Y. Li, and C. Huang, *Privacy-preserving and verifiable deep learning inference based on secret sharing*, Neurocomputing, vol. 483, pp.221-234, 2022.
[21] X. Ma, F. Zhang, X. Chen, and J. Shen, *Privacy preserving multi-party computation delegation for deep learning in cloud computing*, Information Sciences, vol. 459, pp. 103-116, 2018.
[22] O. Nassef, W. Sun, H. Purmehdi, M. Tatipamula, and T. Mahmoodi, *A survey: Distributed Machine Learning for 5G and beyond*, Computer Networks, 108820, ISSN 1389-1286, 2022.
[23] X. Yin, Y. Zhu, and J. Hu, *A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions*, ACM Computing Surveys, vol. 54, no. 6, pp. 1-36, 2022.
[24] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, *A Hybrid Approach to Privacy-Preserving Federated Learning*, Proc. of the 12th ACM Workshop on Artificial Intelligence and Security, pp. 1-11, 2019.
[25] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, *Federated Learning With Differential Privacy: Algorithms and Performance Analysis*, IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3454-3469, 2020.

[26] R. Hu, Y. Guo, and Y. Gong, *Concentrated Differentially Private Federated Learning With Performance Analysis*, IEEE Open Journal of the Computer Society, vol. 2, pp. 276-289, 2021.

[27] J. Gao, B. Hou, X. Guo, Z. Liu, Y. Zhang, K. Chen, and J. Li, *Secure Aggregation is Insecure: Category Inference Attack on Federated Learning*, IEEE Trans. on Dependable and Secure Computing, DOI: 10.1109/TDSC.2021.3128679, 2021.

[28] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, *Federatedlearning for mobile keyboard prediction*, arXiv:1811.03604, 2018.

[29] L. Huang, Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu, *Loadaboost: Loss-based adaboost federated machinelearning on medical data*, arXiv:1811.12629, 2018.

[30] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, *Federated learning for ultra-reliable low-latency v2vcommunications*, 2018 IEEE Global Communications Conference, pp. 1-7, 2018.

[31] Y. Qin, H. Mastutani, and M. Kondo, *A Selective Model Aggregation Approach in Federated Learning for Online Anomaly Detection*, International Conference on Cyber, Physical and Social Computing, pp. 178-185, 2020.

[32] M. von Maltitz, S. Smarzly, H. Kinkelin, and G. Carle, *A management framework for secure multiparty computation in dynamic environments*, Proc. of IEEE/IFIP Network Operations and Management Symposium, pp. 1-7, 2018.

[33] M. Asad, A. Moustafa, and T. Ito, *FedOpt: Towards Communication Efficiency and Privacy Preservation in Federated Learning*, Applied Sciences, vol. 10, no. 8, p. 2864, 2020.

[34] R. Liu, Y. Cao, M. Yoshikawa, and H. Chen, *FedSel: Federated SGD under Local Differential Privacy with Top-k Dimension Selection*, arXiv:2003.10637, 2020.

[35] S. S. Rathna and T. Karthikeyan, *Survey on Recent Algorithms for Privacy Preserving Data mining*, International Journal of Computer Science and Information Technologies, vol. 6, issue 2, pp. 1835-1840, 2015.

[36] J. Yuan and S. Yu, *Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing*, IEEE Trans. On Parallel and Distributed Systems, Vol. 25, issue 1, pp. 212-221, 2013.

[37] N. Schlitter, *A Protocol for Privacy Preserving Neural Network Learning on Horizontal Partitioned Data*, Privacy Statistics in Databases (PSD), 2008.

[38] H. Miyajima, H. Miyajima, and N. Shiratori, *Fast and Secure Back-Propagation Learning using Vertically Partitioned Data with IoT*, CANDAR 2019 : The Seventh International Symposium on Computing and Networking, Nagasaki, November, pp. 450-454, 2019.

[39] 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.

[40] 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.

[41] S. Ruder, *An Overview of Gradient Descent Optimization Algorithms*, http://ruder.io/optimizing-gradient-descent/, 2016 (accessed 14 Mar. 2018).

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

[43] 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.

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

Hirofumi Miyajima received the Ph.D. degree from Kagoshima University, Japan, in 2016. He is currently an Associate Professor at Nagasaki University, Japan. His research interests include soft computing and machine learning.

Noritaka Shigei received the B.S., M.S., and Ph.D. degrees from Kagoshima University, Japan, in 1992, 1997, and 1997, respectively. He is currently an Associate Professor at Kagoshima University, Japan. His research interests include machine learning, information networks, and digital communication systems.

Hiromi Miyajima received the Ph.D. degree from Tohoku University, Japan, in 1979. He was a Professor at Kagoshima University from 1991 to 2016. He is currently an Emeritus Professor with Kagoshima University, Japan. His research interests include communication and network engineering and informatics/intelligent informatics.

Norio Shiratori is currently a Professor at Chuo University, Tokyo, and an Emeritus Professor at Tohoku University, Sendai, Japan. He has published more than 15 books and over 600 refereed papers in computer science and related fields. He is a Life Fellow of the IEEE. He is also a Fellow of the Japan Foundation of Engineering Societies, the Information Processing Society of Japan (IPSJ), and the Institute of Electronics, Information and Communication Engineers (IEICE). He was a former President of IPSJ from 2009 to 2011. He was an IEICE Honorary Member in 2012 and an IPSJ Honorary Member in 2013. He was a recipient of the Minister of MEXT Award from the Japanese Government in 2016, the Science and Technology Award from the Ministry of Education, Culture, Sports, Science, and Technology, in 2009, the IEICE Achievement Award, in 2001, the IEICE Contribution Award, in 2011, the IPSJ Contribution Award, in 2008, the IPSJ Memorial Prize Winning Paper Award, in 1985, the IPSJ Best Paper Award, in 1997, the IEICE Best Paper Award, in 2001, the IEEE 5th SCE01 Best Paper Award, in 2001, the IEEE ICPADS 2000 Best Paper Award, in 2000, and the IEEE 12th ICOIN Best Paper Award.