# Secure Learning Systems using Vertically Partitioned Data with IoT

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

***Abstract*—The use of cloud computing is expanding. However, as the number of terminals (things) in the cloud computing system increases, its capability limit becomes apparent. The limit leads to a significant delay in processing time. Edge (fog) computing is one of the methods for improving conventional cloud systems. The key idea of edge computing is to place edge servers between the cloud servers and the terminals (things). The capacity of each edge may not be so high, but many edges cooperate in executing tasks to achieve high processing power. Then, how should machine learning be realized on the edge system? One of the problems in machine learning is that the confidentiality cannot be maintained because the learning data is concentrated in one place. As a means for solving this, a method for realizing learning while distributed data to multiple servers is being sought. Safe systems using distributed processing have attracted attention. SMC (Secure Multiparty Computation) is one of the typical models. Horizontally partitioning and vertically partitioning data are known as methods of partitioning dataset for SMC. Several methods have been proposed to achieve machine learning in traditional systems. In addition, some machine learning algorithms have been proposed using HPD (horizontally partitioned data) in edge systems. On the other hand, few studies on machine learning using Vertically Partitioned Data (VPD) have been done so far.**

**This paper proposes secure BP (Back-Propagation) neural network learning for VPD. In addition to a BP learning algorithm of SMC for three-layered neural networks, a generalized learning algorithm of SMC for VPD based on SDM (Steepest Descent Method), which covers many learning methods based on SDM, is presented. Further, Neural Gas algorithm based on this generalized learning method is proposed. Numerical simulations show the effectiveness of the proposed methods.**

***Index Terms*—IoT, Machine learning, Secure multiparty computation, Batch processing, Back propagation, Edge computing, Vertically partitioned data, Neural Gas algorithm.**

## I. Introduction

CLOUD computing, which is one of the basic technologies that support ICT, is used in various fields. However, with the transition of things to the Internet of Things (IoT), the number of servers (things) connected to cloud systems is increasing. It is known that this increases the load on the server and significantly reduces the processing power of the cloud system. This causes a significant delay in processing time [1], [2], [3], [4]. In particular, this has fatal consequences in areas that require online or high-speed processing, such as autonomous driving. Edge (or fog) computing systems are known as a way to improve traditional cloud systems. The basic idea is to introduce a system that places an edge (server) between the cloud and the terminal (thing) [1], [2], [3]. In this case, the edge performs normal tasks and the cloud server performs heavy or difficult tasks. As a result, the load on the cloud system can be significantly reduced. In addition, the close distance between the data and the edges allows for faster processing in edge systems. Even if the capacity of each edge is not so high, many edges work together to perform tasks and achieve high processing power. So how do edge systems enable machine learning? The purpose of learning is to find hidden relationships (information) from the collected data. One of the problems with machine learning is that the learning data is concentrated in one place, so confidentiality is maintained. Some methods for realizing machine learning on the cloud and the edge have been proposed [5], [6], [7], [8], [9], [10], [11], [12]. HPD and VPD are known as methods to partition a dataset. The former divides a dataset into subsets (shards), and the latter divides the dataset into element-separated subsets. Also, some algorithms of machine learning using HPD on edge systems have been proposed [9], [11], [16]. On the other hand, few studies have been done on machine learning using VPD [12], [17]. It is desired to construct a highly secure machine learning method using VPD consisting of element-separated subsets.

This paper proposes a fast and secure BP neural network learning on VPD with an edge system. In addition to a BP learning algorithm for three-layered neural networks, a generalized learning algorithm for VPD based on SDM (Steepest Descent Method), which covers many learning methods based on SDM, is presented. Further, Neural Gas algorithm based on this generalized learning method is proposed. Its effectiveness is shown for function approximation, pattern classification, and clustering problems.

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

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

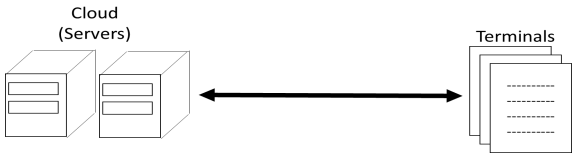
Norio Shiratori is a Professor of Chuo University, 1-13-27, Kasuga, Bunkyoku, Tokyo, Japan (e-mail: norio@riec.tohoku.ac.jp).

## II. Preliminary

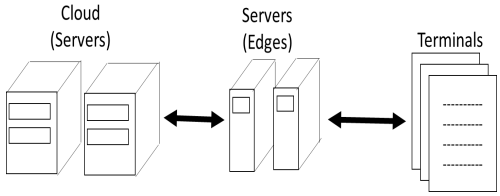
### A. A configuration of the edge computing system

The purpose of edge systems is to combine multiple servers with low capabilities (called edges) to create a high-performance system. High processing power includes 1) high speed of processing, 2) secure computation, 3) short communication time, etc. In the following, we propose a method to efficiently realize machine learning using learning data on the edge system is proposed.

Fig.1(a) shows a conventional cloud system. Fig.1(b) shows an example of an edge computing system. This system consists of a terminal and several edges connected at a close distance from the terminal. Each server (edge) is directly connected to each terminal. The data provided to the terminal is sent to each edge for processing. The challenge is how to share and distribute the data among the servers in order to perform secure and fast computation.




(a) Conventional cloud system




(b) Edge system

Fig. 1. Cloud and Edge systems.

Fig.2 shows the relationship between the edges and terminals of the edge system used in this paper. The edge system consists of $m+1$ servers (edges) and $m$ terminals.

### *B. Steepest descent method in machine learning*

The purpose of machine learning is to provide a way for the system to realize the input / output relationships of given learning data by estimating the parameters of the system. Parameters are usually estimated by sequentially updating the parameters based on SDM. SDM applications include supervised learning such as BP learning and fuzzy modeling, and unsupervised learning such as K-means, NG (Neural Gas), and SOM (Self Organization Mapping). SDM is one of the ways to minimize the target function $J(\boldsymbol{\theta})$ for system parameters. In this case, learning is achieved by updating the system parameters in the opposite direction of the gradient $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ of the objective function. The learning rate $\eta$ determines the size of the step. This method is performed based on the following formula [13].

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

where $\nabla J(\boldsymbol{\theta})$ is the amount of updates calculated based on SDM.

That is, the parameter $\boldsymbol{\theta}$ is updated based on Eq.(1) using learning data to reach a local minimum. There are three ways to use the learning data: online, mini-batch and batch. The following describes the mini-batch method [13].

For the integer $i$, let $Z_i = \{1, 2, \cdots, i\}$ and $Z_i^* = \{0, 1, \cdots, i\}$. Let $D$ be the set of learning data $|D| = L$. The set $D$ consists of $Q$ subsets such as $D = \bigcup_{l=1}^{Q} B_l$ and $B_i \cap B_j = \emptyset$ for $i \neq j$, where $|B_l| = b_l$ for $l \in Z_Q$ and $L = \sum_{l=1}^{Q} b_l$. Let $t = 1$.

Step 1: Given $l \in Z_Q$, then the set $B_l$ is given.

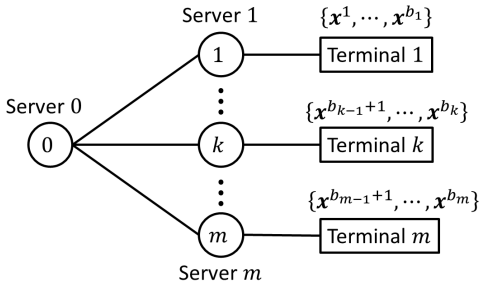
Step 2: Update $\boldsymbol{\theta}$ based on Eq.(1) using $B_l$.

Step 3: If $\nabla J(\boldsymbol{\theta})$ is not sufficiently small, then go to Step 1 with $t \leftarrow t+1$. Otherwise, the algorithm terminates.

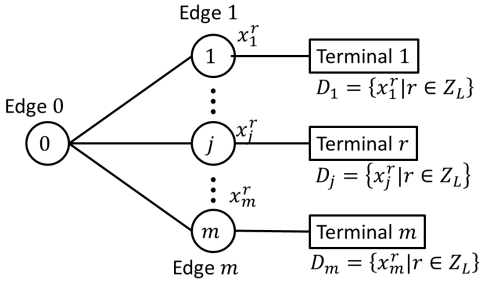
For $Q = 1$ and $Q = L$, this method is called online method and batch method, respectively.

### *C. System configuration using securely shared data*

Let us consider conventional works with securely shared data. In order to solve the problem, three partitioned representations of data such as horizontally, vertically and any




(a) Horizontally partitioned data : The dataset $\{\boldsymbol{x}^{b_{k-1}+1}, \cdots, \boldsymbol{x}^{b_k}\}$ provided to the $k$-th terminal is set to the $k$-th Server for $k \in Z_m$.




(b) Vertically partitioned data : Each element $x_k^r$ of data $\boldsymbol{x}^r$ for $r \in Z_L$ is sent to the $k$-th Server.

Fig. 2. Horizontally and vertically partitioned data.

partitioned data are known [5], [6], [7]. Let us explain HPD and VPD using an example of Table I. In Table I, a and b are original data (marks) and ID is a student identifier. The purpose of computation is to get the averages of them.

First, let us explain VPD [7]. All the data are shared into two servers, Server 1 and 2 as follows:

Server 1: data for ID=1, 2, 3,

Server 2: data for ID=4,5.

In Server 1, each average for A or B is computed as $(85 + 59 + 26)/3$ and $(30 + 34 + 20)/3$, respectively. In Server 2, each average for A or B is computed as $(77 + 46)/2$ and $(67 + 48)/2$, respectively. As a result, two averages for subjects A and B are 58.6 and 39.8, respectively. Each server cannot know half of the dataset, so security is maintained.

Next, let us consider VPD. All the data are shared into two servers, Server 1 and 2, as follows:

Server 1: data for subject A,

Server 2: data for subject B.

In this case, two averages for subjects A and B are computed as 58.6 and 39.8 as usual, respectively. Each server can know only data for subject A or B, so security is maintained.

Let us explain VPD and HPD for the edge system. In HPD, each data provided from any terminal is set to each server as it is (Fig.2(a)). In VPD, any data is sent separately to each Server. For example, the $k$-th element $x_k^r$ of the $r$-th data $\boldsymbol{x}^r = (x_1^r, \cdots, x_k^r, \cdots, x_m^r)$ is sent to the $k$-th Server (Fig.2(b)).

Several methods have been proposed to achieve machine learning in traditional systems. In addition, some machine learning algorithms have been proposed using horizontally

TABLE I
CONCEPT OF HORIZONTALLY AND VERTICALLY PARTITIONED DATA COMPOSED OF TWO SERVERS.

| | | Server 1 | Server 2 | |
|---|---|---|---|---|
| | ID | Subset A: A | Subset B: B | average |
| Server 1 | 1 | 85 | 30 | 57.5 |
| | 2 | 59 | 34 | 46.5 |
| | 3 | 26 | 20 | 23 |
| Server 2 | 4 | 77 | 67 | 72 |
| | 5 | 46 | 48 | 47 |
| | average | 58.6 | 39.8 | 49.2 |

Horizontally partitioned data

Vertically partitioned method

partitioned data in edge systems. For example, FL is one of them. In the following, secure BP neural network learning on VPD with edge system (for IoT) and the general learning algorithm for VPD based on SDM are proposed.

### *D. BP neural network learning*

In the following, let us explain a three-layered neural network as an example of BP neural network learning without loss of generality. Let $\boldsymbol{x}^l \in J^N$ for $l \in Z_L$ and $d : J^N \to J$, where $J = [0,1]$ or $\{-1,1\}$. For the sets $D = \{(\boldsymbol{x}^l, d(\boldsymbol{x}^l)) | \boldsymbol{x}^l \in J^N, l \in Z_L\}$, $D_{in} = \{\boldsymbol{x}^l | l \in Z_L\}$ and $D_{out} = \{d(\boldsymbol{x}^l) | l \in Z_L\}$ of learning data, let us determine parameters of the three-layered neural network identifying learning data by BP method, where $d(\boldsymbol{x}^l)$ means the desirable output for input data $\boldsymbol{x}^l$. Let $h = h_2 \circ \boldsymbol{h}_1 : J^N \to J$ for $\boldsymbol{h}_1 : J^N \to J^M$ and $h_2 : J^M \to J$. Let $N$ and $M$ be the numbers of elements for the first and second layers, respectively. Let $\boldsymbol{w}_j$ for $j \in Z_M^*$ and $\boldsymbol{v} = (v_0, v_1, \cdots, v_L)$ be weights for the second and output layers, respectively. Then $\boldsymbol{h}_1$ and $h_2$ are as follows (See Fig.3):

$$\begin{aligned} y_i &= h_{1i}(\boldsymbol{x}) = \tau\left(\sum_{j=0}^{N} w_{ij} x_j\right), \quad (2) \\ x_0 &= 1, \ \boldsymbol{h}_1 = (h_{11}, \cdots, h_{1i}, \cdots, h_{1M}) \\ \tau(u) &= \frac{1}{1+\exp(-u)} \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{x} &= (x_0, x_1, \cdots, x_j, \cdots, x_N) \in J^N \\ \boldsymbol{y} &= (y_0, y_1, \cdots, y_i, \cdots, y_M) \in J^M \end{aligned}$$

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

Further,

$$\begin{aligned} h_2(\boldsymbol{y}) &= \tau\left(\sum_{i=0}^{M} v_i y_i\right), \quad (3) \\ y_0 &= 1, \end{aligned}$$

where $v_0$ means the threshold value.

Then, the evaluation function is defined as follow:

$$E = \frac{1}{2L} \sum_{l=1}^{L} \left(h(\boldsymbol{x}^l) - d(\boldsymbol{x}^l)\right)^2 \quad (4)$$

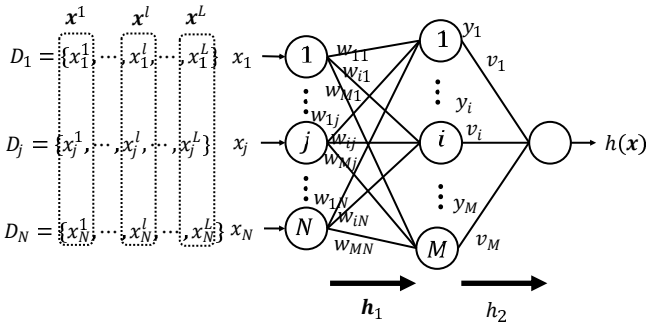



Fig. 3. The relation between the dataset and three-layered neural network: Each element $\boldsymbol{x}_j^l$ of the set $D_j$ for $j \in Z_N$ is provided to the $j$-th element of the first layer.

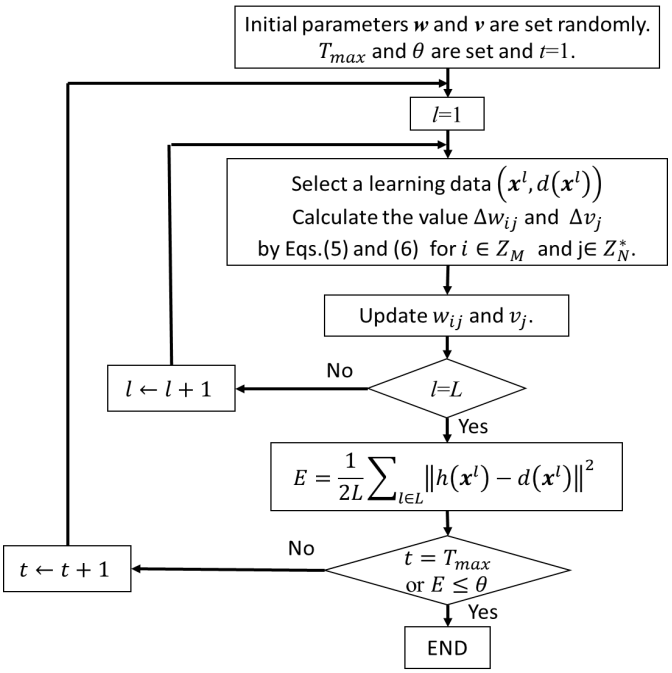



Fig. 4. The conventional BP learning (online type). For batch learning, Eq.(9) and (10) are used instead of Eqs.(5) and (6), respectively.

The weights $\boldsymbol{w}$ and $\boldsymbol{v}$ are updated based on the BP method of the online type as follows[14] :

$$\begin{aligned} \triangle v_i(\boldsymbol{x}^l) &= -\alpha_1 \delta_2(\boldsymbol{x}^l) h_{1i}(\boldsymbol{x}^l) \quad (5) \\ \triangle w_{ij}(\boldsymbol{x}^l) &= -\alpha_2 \delta_{1i}(\boldsymbol{x}^l) x_j^l \text{ for } j \in Z_N^* \text{ and } i \in Z_M \quad (6) \end{aligned}$$

where $\alpha_1$ and $\alpha_2$ are learning coefficients,

$$\delta_2(\boldsymbol{x}) = (h(\boldsymbol{x}) - d(\boldsymbol{x}))h(\boldsymbol{x})(1 - h(\boldsymbol{x})) \quad (7)$$

and

$$\delta_{1i}(\boldsymbol{x}) = \delta_2(\boldsymbol{x}) v_i h_{1i}(\boldsymbol{x})(1 - h_{1i}(\boldsymbol{x})). \quad (8)$$

Then, the BP learning method is shown in Fig.4 [14].

In batch learning, the following equations are used instead of Eqs.(5) and (6).

$$\begin{aligned} \triangle v_i &= -\sum_{l=1}^{L} \triangle v_i(\boldsymbol{x}^l) \text{ for } i \in Z_M^* \quad (9) \\ \triangle w_{ij} &= -\sum_{l=1}^{L} \triangle w_{ij}(\boldsymbol{x}^l) \text{ for } j \in Z_N^* \text{ and } i \in Z_M \quad (10) \end{aligned}$$

## III. Batch BP learning for secure multiparty computation with IoT

### A. Batch BP learning for three-layered neural network for SMC

In this section, a fast and secure BP neural network learning on VPD with the edge system is proposed (See Fig.2(b)). $L$ pieces of learning data are given to Terminals. Each element of any learning data is assigned to each Edge. The $j$-th Edge has the set $D_j = \{x_j^1, \cdots, x_j^L\}$ for $j \in Z_m$. The key of the proposed method is to use VPD and batch learning. The problem is how weights $w_{ij}$ and $v_i$ are updated. Let $m = N$. Let $D_j = \{x_j^1, \cdots, x_j^L\}$ for $j \in Z_N$ and $D_{in} = \{\boldsymbol{x}^l | l \in Z_L\}$ and $D_{out} = \{d(\boldsymbol{x}^l) | \boldsymbol{x}^l \in D_{in}\}$. In the proposed method with VPD, each set $D_j$ is given as shown in Fig.3. Eq.(2) is rewritten as follows :

$$\begin{aligned} h_{1i}(\boldsymbol{x}^l) &= \tau\left(w_{i0} + \sum_{j=1}^{N} w_{ij} x_j^l\right) \\ &= \tau\left(w_{i0} + \sum_{j=1}^{N} s_{ij}^l\right) \end{aligned} \tag{11}$$

where $s_{ij}^l = w_{ij} x_j^l$ for $i \in Z_M$ and $l \in Z_L$. The term $s_{ij}^l$ for $i \in Z_M$ and $l \in Z_L$ is computed using the set $D_j$ and $w_{ij}$ at Edge $j$. The result for each Edge is sent to Edge 0, and an output of network $h(\boldsymbol{x}^l)$ is computed. Then, any Edge can know only an element of each data $\boldsymbol{x}$. That is, the secure computation is performed. Further, each computation on Edge is done simultaneously (in parallel), so fast computation is performed.

The proposed method for the three-layered neural network is shown in Table II. Terminals collect learning data, and each terminal has the set $D_j$. Each element of any data $\boldsymbol{x} \in R^N$ is sent to each Edge (See Fig.2(b)), where $R$ is the set of all real numbers. As the initial condition for learning data and parameters, Edge 0 has the weight $\boldsymbol{v}$ and the set $D_{out}$, and each Edge $j$ has the set $D_j$ and the weight $w_{ij}$ for $i \in Z_M$. At Step 1, Edge $j$ calculates the weighted input $s_{ij}^l$ for $i \in Z_M$ and $l \in Z_L$, and the result is sent to Edge 0. At Step 2 of Edge 0, $h_{1i}(\boldsymbol{x}^l) = \tau\left(w_{i0} + \sum_{j=1}^{N} s_{ij}^l\right)$ for $i \in Z_M$ is calculated using $s_{ij}^l$ for $j \in Z_M$, and $h(\boldsymbol{x}^l)$ for $l \in Z_L$ is calculated using $h_{1i}(\boldsymbol{x}^l)$. Further, the errors $\delta_2(\boldsymbol{x}^l)$ and $\delta_{1i}(\boldsymbol{x}^l)$ for each data $\boldsymbol{x}^l \in D$ are calculated, and the weight $\boldsymbol{v}$ is updated based on Eq.(9). The weight $w_{i0}$ for $i \in Z_M$ is updated. The error $\delta_{1i}(\boldsymbol{x}^l)$ for $l \in Z_M$ is sent to all Edges. At Step 3 of Edge $j$, the updated amount $\triangle w_{ij}$ is calculated and the weight $w_{ij}$ is updated. Further, $s_{ij}^l$ for $i \in Z_M$ and $l \in Z_L$ is calculated and is sent to Edge 0. At Step 4 of Edge 0, $h_{1i}(\boldsymbol{x}^l)$ and $h(\boldsymbol{x}^l)$ for $l \in Z_L$ are calculated. Further, the difference between output $h(\boldsymbol{x})$ and desirable output $d(\boldsymbol{x})$ is calculated and is evaluated. If the error $E(t)$ is sufficiently small, then the algorithm terminates. Otherwise, go to Step 2 with $t \leftarrow t+1$.

The proposed method in Table II can be rewritten into a simplified BP learning method by performing the following replacement. That is, Step 2 in Edge 0 is replaced as follows:

**Step 2:** Calculate $h_{1i}(\boldsymbol{x}^l) = \tau(\sum_{j=0}^{N} s_{ij}^l)$ and $h(\boldsymbol{x}^l) = \tau(\sum_{i=0}^{M} v_i h_{1i}(\boldsymbol{x}^l))$ for $l \in Z_L$. Calculate $p_{ij} = \sum_{l=1}^{L} \delta_{1i}(\boldsymbol{x}^l) s_{ij}^l$ for $i \in Z_M$ and $j \in Z_N$, and send them to each Edge. Calculate $\triangle w_{i0} = -\sum_{l \in Z_L} \alpha_1 \delta_{1i}(\boldsymbol{x}^l)$ and $\triangle v_i$ of Eq.(10) and $v_i \leftarrow v_i + \triangle v_i$ and $w_{i0} \leftarrow w_{i0} + \triangle w_{i0}$ for $i \in Z_M$.

Further, Step 3 in Edge $j$ is replaced as follows:

**Step 3:** Calculate $w_{ij} \leftarrow w_{ij} + \alpha_1 p_{ij}/w_{ij}$ and $s_{ij}^l = w_{ij} x_j^l$ for $i \in Z_M$ and $l \in Z_L$, and send them to Edge 0.

This result is shown in Ref.[17].

### B. The generalized algorithm for VPD based on SDM

In the previous section, a distributed calculation for BP learning based on SDM was shown using the data structure $D_j$ for $j \in Z_N$. In this section, we generalize this result and present a generalized algorithm for VPD based on SDM. The idea is to learning by partitioning the parameters into those that can be divided and those that cannot. The generalized algorithm represents many learning methods based on SDM. For example, the algorithm represents supervised learning such as multi-layered neural network and fuzzy modeling and unsupervised learning such as NG and k-means.

Let us explain the proposed method. The set $D$ of learning data is decomposed into set $D_j$ for $j \in Z_N$ based on VPD. At the same time, the sets $P$ and $Q$ of parameters are given, where the union $U = P \cup Q$ is the set of all the parameters, $P = \cup_{j=1}^{N} P_j$ and $Q$ is the set of parameters in $U$ that are not in $P$. First, the set $Q$ and the set $D_{out}$ are given to Edge 0. The set $P_j$ of parameters and the set $D_j$ for $j \in Z_N$ are given to Edge $j$. In Step 1, partial calculation $q_{ij}^l$ is performed at Edge $j$ using $P_j$ and $D_j$, and the result is sent to Edge 0. In Step 2, the updated amount $\delta(\boldsymbol{x}^l)$ for $l \in Z_L$ is calculated on Edge 0 by using $D_{out}$ and $q_{ij}^l$ and send them to Edge $j$. Further, each parameter of the set $Q$ is updated. In Step 3, at Edge $j$, each parameter $p_{ij}$ of $P_j$ is updated using the updated amount $\triangle p_{ij} = \sum_{l \in Z_L} \alpha_2 \delta(\boldsymbol{x}^l) x_j^l$. Further, a partial calculation $q_{ij}^l$ is performed, and sent to Edge 0. In Step 4, it is determined whether the learning result is satisfied with the condition of learning. If not, return to Step 2 and repeat the same processes. The generalized learning algorithm for VPD based on SDM is shown in Table III.

It will be clear that Table II is a special case of Table III.

### C. A secure NG based on the generalized learning algorithm

In the following, NG (including k-means) is shown as an example of unsupervised learning based on the generalized learning. In this case, $D_{out} = \phi$, $Q = \phi$ and $P = W$ hold. First, NG will be explained.

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 $d$ and $r$ are positive integers.

Let the winner vector $\boldsymbol{w}_{i(\boldsymbol{x})}$ be defined for any vector $\boldsymbol{x} \in X$ as follows:

$$i(\boldsymbol{x}) = \arg\min_{i \in Z_r} ||\boldsymbol{x} - \boldsymbol{w}_i|| \tag{12}$$

From the finite set $W$, $X$ is partitioned as follows:

$$X_i = \{\boldsymbol{x} \in X | ||\boldsymbol{x} - \boldsymbol{w}_i|| \leq ||\boldsymbol{x} - \boldsymbol{w}_j|| \ for \ j \in Z_r\} \tag{13}$$

The sets $X$ and $W$ are called sets of input and reference vectors, respectively.

TABLE II
THE PROPOSED METHOD FOR BP NEURAL NETWORK LEARNING.

| | Edge 0 | Edge $j$ |
|---|---|---|
| Initial condition | The weight $\boldsymbol{v} = (v_0, v_1, \cdots, v_M)$ and $w_{i0}$ for $i \in Z_M$ are selected randomly. $D_{out}$, $\alpha_1$, $T_{max}$ and $\theta$ are given. Set $t = 1$. | The set $D_j$ is given. The weight $w_{ij}$ for $i \in Z_M$ is selected randomly. The constant $\alpha_2$ is given. |
| Step 1 | | Calculate $s_{ij}^l = w_{ij} x_j^l$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. |
| Step 2 | Calculate $h_{1i}(\boldsymbol{x}^l) = \tau(w_{i0} + \sum_{j=1}^N s_{ij}^l)$ and $h(\boldsymbol{x}^l) = \tau(v_0 + \sum_{i=1}^M v_i h_{1i}(\boldsymbol{x}^l))$ for $l \in Z_L$. Calculate $\delta_2(\boldsymbol{x}^l)$ and $\delta_{1i}(\boldsymbol{x}^l)$ for $i \in Z_M$ and $\triangle v_i$ of Eq.(9) and $v_i \leftarrow v_i + \triangle v_i$. $\delta_{1i}(\boldsymbol{x}^l)$ for $i \in Z_M$ and $l \in Z_L$ is sent to each Edge. Calculate $\triangle w_{i0} = -\sum_{l \in Z_L} \alpha_1 \delta_{1i}(\boldsymbol{x}^l)$ and $w_{i0} \leftarrow w_{i0} + \triangle w_{i0}$ for $i \in Z_M$. | |
| Step 3 | | Calculate $\triangle w_{ij} = -\sum_{l \in Z_L} \alpha_2 \delta_{1i}(\boldsymbol{x}^l) x_j^l$ and $w_{ij} \leftarrow w_{ij} + \triangle w_{ij}$. Calculate $s_{ij}^l = w_{ij} x_j^l$ for $i \in Z_M$ and $l \in Z_L$ and send them to Edge 0. |
| Step 4 | Calculate $h_{1i}(\boldsymbol{x}^l) = \tau(w_{i0} + \sum_{j=1}^N s_{ij}^l)$ and $h(\boldsymbol{x}^l) = \tau(v_0 + \sum_{i=1}^M v_i h_{1i}(\boldsymbol{x}^l))$ for $l \in Z_L$ and $E(t) = \frac{1}{2L} \sum_{l=1}^L (h(\boldsymbol{x}^l) - d(\boldsymbol{x}^l))^2$. If $E(t) < \theta$ or $t \geq T_{max}$ then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$ | |

TABLE III
THE GENERALIZED LEARNING ALGORITHM FOR VPD BASED ON SDM.

| | Edge 0 | Edge $j$ |
|---|---|---|
| Initial condition | Each parameter of the set $Q$ is selected randomly. $D_{out}$ and $T_{max}$ are given. Set $t = 1$. | Each parameter $p_{ij}$ of the set $P_j$ is selected randomly. The constant $\alpha_2$ is given. The set $D_j$ is given. |
| Step 1 | | Calculate $q_{ij}^l = f_j(p_{ij}, x_j^l)$ for $l \in Z_L$ and $i \in Z_M$ and send them to Edge 0. |
| Step 2 | Calculate the update amount $\delta(\boldsymbol{x}^l)$ for $l \in Z_L$ by using $q_{ij}^l$ and $D_{out}$ and send them to each Edge. Update each parameter of the set $Q$. | |
| Step 3 | | Calculate $\triangle p_{ij} = -\sum_{l \in Z_L} \alpha_2 \delta(\boldsymbol{x}^l) x_j^l$ for $l \in Z_L$ and update $p_{ij} \leftarrow p_{ij} + \triangle p_{ij}$. Calculate $q_{ij}^l = f_j(p_{ij}, x_j^l)$ for $l \in Z_L$ and send them to the Edge 0. |
| Step 4 | Calculate MSE by using by using $q_{ij}^l$ for $j \in Z_N$. If MSE is sufficiently small or $t \geq T_{max}$ then the algorithm terminates else go to Step 2 with $t \leftarrow t + 1$. | |

TABLE IV
A SECURE NG ALGORITHM BASED ON THE GENERALIZED ALGORITHM.

| | Edge 0 | Edge $j$ |
|---|---|---|
| Initial condition | $T_{max}$, $\theta$ and $\alpha$ are given. Set $t = 1$. | The set $D_j$ is given. The weight $w_{ij}$ for $i \in Z_r$ is selected randomly. The constant $\alpha$ is given. |
| Step 1 | | Calculate $q_{ij}^l = w_{ij} - x_j^l$ for $l \in Z_L$ and $i \in Z_r$ and send them to Edge 0. |
| Step 2 | Calculate $d(\boldsymbol{x}^l, \boldsymbol{w}_i) = \sum_{j=1}^m q_{ij}^l$ for $l \in Z_L$. The rank $k_i(\boldsymbol{x}^l, \boldsymbol{w}_i)$ is determined using $d(\boldsymbol{x}^l, \boldsymbol{w}_i)$ and send them to each Edge. | |
| Step 3 | | Calculate $\triangle w_{ij} = \alpha \sum_{l \in Z_L} h_\lambda(k_i(\boldsymbol{x}^l, \boldsymbol{w}_i)) q_{ij}^l$ for $i \in Z_r$ and $w_{ij} \leftarrow w_{ij} + \triangle w_{ij}$ for $i \in Z_r$. Calculate $q_{ij}^l = w_{ij} - x_j^l$ for $l \in Z_L$ and $i \in Z_r$ and send them to Edge 0. |
| Step 4 | If $t > T_{max}$ them algorithm terminates else go to Step 2 with $t \leftarrow t + 1$. | |

For NG method, the following method is used[11], [15] :

Given an input vector $\boldsymbol{x}$, we determine the neighborhood-ranking $\boldsymbol{w}_{i_k}$ for $k \in Z_{r-1}^*$, being the reference vector for which there are $k$ vectors $\boldsymbol{w}_j$ with

$$||\boldsymbol{x} - \boldsymbol{w}_j|| < ||\boldsymbol{x} - \boldsymbol{w}_{i_k}|| \tag{14}$$

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

If the number $k$ associated with each vector $\boldsymbol{w}_i$ is denoted by $k_i(\boldsymbol{x}, \boldsymbol{w}_i)$, then the adaption step for adjusting the $\boldsymbol{w}_i$'s is given by

$$\triangle \boldsymbol{w}_i = \alpha h_\lambda(k_i(\boldsymbol{x}, \boldsymbol{w}_i))(\boldsymbol{x} - \boldsymbol{w}_i) \tag{15}$$

$$\triangle w_{ij} = \alpha h_\lambda(k_i(\boldsymbol{x}, \boldsymbol{w}_i))(x_j - w_{ij}) \tag{16}$$

$$h_\lambda(k_i(\boldsymbol{x}, \boldsymbol{w}_i)) = \exp\left(-k_i(\boldsymbol{x}, \boldsymbol{w}_i)/\lambda\right) \tag{17}$$

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

where $T_{max}$, $\alpha_{int}$ and $\alpha_{fin}$ mean the maximum number of learning time and the real numbers. The constant $\lambda$ is called decay one. If $\lambda \to 0$, Eq.(15) becomes equivalent to k-means method. Eq.(15) is written as Eq.(16) in the element-wise form.

**Learning Algorithm of Neural Gas [15]**

Input : The set $D_{in}\,(= X)$ of data

Output : The set $W$ of reference vectors

**Step 1:** The initial values of reference vectors are set

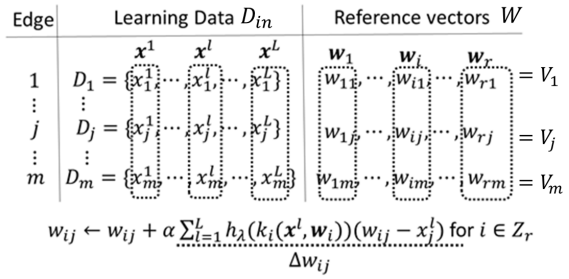



Fig. 5. The relation between learning data and parameters for the proposed NG.

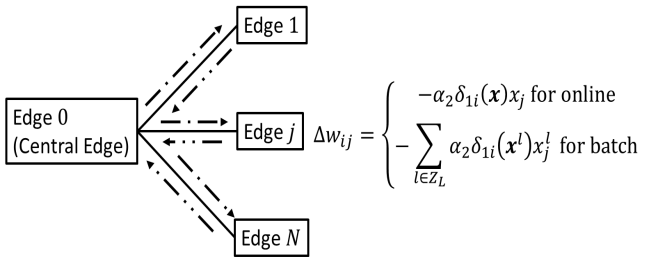



Fig. 6. Online and Batch Processing : The arrows of one-dot and two-dot chain lines mean to process in parallel each task. In online case, each weight $w_{ij}$ for $i \in Z_M$ is updated for each data $\boldsymbol{x}$ at Edge $j$. In batch case, each weight $w_{ij}$ is updated for all data $\boldsymbol{x}^l$ ($l \in Z_L$) and the batch step is performed at a time in the proposed method.

randomly. The learning coefficients $\varepsilon_{int}$ and $\varepsilon_{fin}$ are set, respectively. Let $T_{max}$ and $\theta$ be the maximum number of learning time and the threshold, respectively.

**Step 2:** Let $t = 1$.

**Step 3:** Give a data $\boldsymbol{x} \in D_{in}$ randomly and neighborhood-ranking $k_i(\boldsymbol{x}, \boldsymbol{w}_i)$ is determined for each $\boldsymbol{w}_i \in W$.

**Step 4:** Each reference vector $\boldsymbol{w}_i$ for $i \in Z_r$ is updated based on Eq.(15)

**Step 5:** If $t \geq T_{max}$, then the algorithm terminates and the set $W = \{\boldsymbol{w}_i | i \in Z_r\}$ of reference vectors is obtained. Otherwise go to Step 3 as $t \leftarrow t + 1$.

Table IV shows a secure NG learning based on Table III. In this case, the relation among the sets $D_j$, $V_j$, and $W$ for $j \in Z_m$ are shown as Fig.5, where $D_j = \{x_j^r | r \in Z_L\}$ and $V_j = \{w_{ij} | i \in Z_r\}$. Further, Eq.(18) is used instead of Eq.(16).

$$\triangle w_{ij} = \sum_{l \in Z_L} \alpha h_\lambda(k_i(\boldsymbol{x}^l, \boldsymbol{w}_i))(x_j^l - w_{ij}) \tag{18}$$

At Step 1 of Edge $j$, the difference $q_{ij}^l$ between the weight $w_{ij}$ for $i \in Z_r$, and the $l$-th input data $x_j^l$ is calculated. At Step 2 of Edge 0, the distance $d(\boldsymbol{x}^l, \boldsymbol{w}_i)$ of input data $\boldsymbol{x}^l$ and the weight $\boldsymbol{w}_i$ for $i \in Z_r$ is calculated, the rank $k_i(\boldsymbol{x}^l, \boldsymbol{w}_i)$ for $i \in Z_r$ is determined, and is sent to each Edge. At Step 3 of Edge $j$, the update amount $\triangle w_{ij} = \alpha \sum_{l=1}^{L} h_\lambda(k_i(\boldsymbol{x}^l, \boldsymbol{w}_i)) q_{ij}^l$ for $i \in Z_r$ is calculated, and the weight $w_{ij}$ is updated. Further, $q_{ij}^l = w_{ij} - x_j^l$ for $l \in Z_L$ and $i \in Z_r$ is calculated and is sent to each Edge. At Step 4 of Edge 0, it is checked if the number of learning times is sufficient.

### D. Why is batch processing effective for IoT?

Let us explain the reason for using BP learning as shown in section III.A. Let create the set $D_j = \{x_j^1, \cdots, x_j^L\}$ for $j \in Z_N$ from the set $D$.

Let us consider the case of online processing using Table II. In Step 1, a data $\boldsymbol{x}$ is selected randomly, $s_{ij} = w_{ij} x_j$ for $i \in Z_M$ is calculated at Edge $j$ and sent to Edge 0. In Edge 0 at Step 2, $h_{1i}(\boldsymbol{x})$ and $h(\boldsymbol{x})$ are calculated. Further, $\delta_2(\boldsymbol{x})$ and $\delta_{1i}(\boldsymbol{x})$ are calculated, and $v_i$ and $w_{i0}$ for $i \in Z_M$ are updated. The result $\delta_{1i}(\boldsymbol{x})$ for $i \in Z_M$ is sent to each Edge. At Step 3, $w_{ij}$ for $i \in Z_M$ is updated at Edge $j$ and the weighted input $s_{ij} = w_{ij} x_j$ for $i \in Z_M$ is calculated using new weight $w_{ij}$, and sent to Edge 0. In Edge 0 at Step 4, if the number of learning times is equal to $L$, then the difference between $h(\boldsymbol{x}^l)$ and $d(\boldsymbol{x}^l)$ for $l \in Z_L$ is evaluated. Otherwise, go to Step 2.

Let us consider the case of batch processing of the proposed method. In this case, the computation of $s_{ij}^l$ for $l \in Z_L$ is done with all elements of $D_j$ as shown in Table II. In Edge $j$ at Step 3, the weight $w_{ij}$ is updated with all elements of $D_j$ at a time. That is, the difference between online and batch processing is to use one data or the set $D_j$ at a time in updating step of the weight. Especially, all parameters $w_{ij}$'s for $i \in Z_N$ are updated at a time in each Edge in the proposed method. Therefore, the proposed method is faster than the online processing (See Fig.6).

## IV. Numerical Simulations for the Proposed Algorithm

In this section, numerical simulations for function approximation, pattern classification and clustering are performed.

### A. Function Approximation for the proposed BP

This simulation uses four systems specified by the following functions with $[0, 1]^4$ (for Eqs.(19) and (20)) and $[-1, 1]^4$ (for Eqs.(21) and (22)). The simulation conditions are $K_w = 0.01$, $K_v = 0.01$, and $T_{max} = 10^6$ for each method. Further, the initial values $w_{ij}$ and $v_i$ are set to randomly on $[0, 1]$, respectively. The numbers of learning and test data randomly selected are 1000 and 1000, respectively.

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{12} \tag{19}$$

$$y = \frac{\sin(2\pi x_1) \times \cos(x_2) \times \sin(\pi x_3) \times x_4 + 1.0}{2.0} \tag{20}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{446.52} \tag{21}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(3e^{3x_3} + 2e^{-4x_4})^{-0.5} - 0.077}{4.68} \tag{22}$$

Table V shows the results of the comparison of accuracy for each method. In each box of Table V, Learn and Test mean MSE for learning and test ($\times 10^{-4}$), respectively. The BP and Batch methods mean the conventional BP and Batch methods without sharing (or partitioning) data, respectively. The Proposed means the proposed method of section III.A. After learning, we compared the conventional and proposed

TABLE V
SIMULATION RESULT OF FUNCTION APPROXIMATION FOR BP

| | | Eq.(19) | Eq.(20) | Eq.(21) | Eq.(22) |
|---|---|---|---|---|---|
| BP | Learn | 1.54 | 26.32 | 1.71 | 4.69 |
| | Test | 1.72 | 29.77 | 1.74 | 5.44 |
| Batch | Learn | 0.34 | 10.99 | 0.28 | 0.76 |
| | Test | 0.46 | 13.23 | 0.32 | 0.98 |
| Proposed | Learn | 0.35 | 10.40 | 0.24 | 0.81 |
| ($m = 4$) | Test | 0.43 | 11.92 | 0.29 | 1.00 |

TABLE VI
THE DATASET FOR PATTERN CLASSIFICATION AND CLUSTERING

| | Iris | Wine | Sonar | BCW | Spam | Skin Seg. |
|---|---|---|---|---|---|---|
| # data | 150 | 178 | 208 | 683 | 4601 | 245057 |
| # input | 4 | 13 | 60 | 9 | 57 | 3 |
| # class | 3 | 3 | 2 | 2 | 2 | 2 |

BP methods in terms of MSE. The result of the simulation is the average value from twenty trials.

The result shows that the accuracy for each method is almost the same. That is, while the proposed method maintains the accuracy of the calculation, it also maintains security.

### *B. Pattern Classification for the proposed BP*

Let us show the result for pattern classification using benchmark problems of Iris, Wine, Sonar, BCW, Spam and Skin Segmentation in the UCI database[18] as shown in Table VI. In Table VI, #data means the number of data. In this simulation, 5-fold cross-validation as an evaluation method is used. Table VII shows the results of the comparison between the conventional and the proposed methods. The BP and Batch methods mean the conventional BP and Batch methods without sharing (or partitioning) data, respectively. The Proposed means the proposed method of section III.A. In each box of Table VII, Learn and Test mean the rate (%) of misclassified data for learning and test, respectively. Each value is average from twenty trials.

The simulation results show that the proposed method is comparable in accuracy with the online and batch learning of the conventional model. Since it is difficult to implement and directly compare the calculation speed, let us consider the difference between the calculation times of the conventional and the proposed methods. Assume that additional time for batch learning and the communication time for the edge system is too short for simplicity. The conventional model using one server takes $L\times M\times N$ and $M\times N$ times for computation on updating of $w_{ij}$'s for the online and batch learning, respectively (See Fig.4).

The proposed model using $N+1$ servers takes $L\times M$ and $N$ times for computation on updating of $w_{ij}$'s for the online and batch learning, respectively (See Table II), because the computation at $N$ edges is performed at a time (in parallel). That is, it is desired the proposed method is $N$ times faster than conventional learning concerning the computation of $w_{ij}$.

### *C. Pattern Clustering for the proposed NG*

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 [18], using the conventional NG methods 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 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 MSE for the Eq.(23) and the global purity (GP) for the Eq.(24).

$$E = \frac{1}{|D|\times|W|}\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{w}\in W} h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}))||\boldsymbol{x}-\boldsymbol{w}||^2 \quad (23)$$

$$GP = \frac{1}{L}\sum_{i\in Z_r}\max_{j\in Z_r}(L_{i,j})\times 100, \quad (24)$$

where $X$ is dataset and $W$ is the set of reference vectors and $h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}))$ is calculated as Eq.(17) and $L_{i,j}$ is the number of data belonging to the $i$-th cluster and the $j$-th actual class and the evaluation function.

Conventional online, batch and proposed NG are abbreviated as Online, Batch and Proposed, respectively. GP and MSE are the GP value (%) and the value of Eq.(23). The values in Table VIII 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.

## V. CONCLUSION

In this paper, secure BP neural network learning on VPD with edge system and the generalized learning algorithm for VPD based on SDM were proposed. Further, a Neural Gas algorithm based on the generalized learning method is proposed. The effectiveness is shown in numerical simulations. In Section II, edge computing systems and a secure data sharing mechanism used in this paper were explained. Further, SDM and the conventional BP method were introduced. In Section III, a fast and secured BP learning method for VPD with the edge system was proposed. Further, the general learning algorithm for VPD based on SDM was proposed. Furthermore, BP and NG methods based on the general learning algorithms were presented. In Section IV, numerical simulations were performed to show the performance of the proposed methods. Generally speaking, it was as follows: It was shown using an edge system composed of $N+1$ edges. Learning data were shared to $N$ pieces of element-separated subsets for $N$ edges, learning was performed simultaneously in each edge and system parameters were updated in each edge using their results. The processing was iterated until satisfactory results were obtained. That is, it was shown that "vertically partitioned data + batch learning (in parallel) = fast and secure computation for edge system". The effectiveness of the idea was shown using BP and NG algorithms. In future works, other applications for SDM and another method for data sharing will be considered.

## 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.

TABLE VII
SIMULATION RESULT OF PATTERN CLASSIFICATION FOR BP

| | | Iris | Wine | Sonar | BCW | Spam | Skin Seg. |
|---|---|---|---|---|---|---|---|
| BP | Learn | 3.18 | 8.23 | 10.88 | 2.38 | 10.10 | 4.92 |
| | Test | 3.47 | 10.03 | 19.07 | 2.81 | 10.24 | 4.92 |
| Batch | Learn | 3.52 | 1.88 | 1.15 | 2.34 | 3.93 | 2.12 |
| | Test | 3.90 | 5.83 | 16.83 | 2.93 | 6.73 | 2.12 |
| Proposed | Learn | 3.53 | 1.92 | 1.27 | 2.34 | 3.90 | 2.17 |
| $(m = \#input)$ | Test | 3.73 | 5.56 | 16.10 | 2.86 | 6.79 | 2.17 |

TABLE VIII
SIMULATION RESULT OF PATTERN CLUSTERING FOR NG

| | | Iris | Wine | Sonar | BCW | Spam |
|---|---|---|---|---|---|---|
| Online | MSE | 6.46 | 47.65 | 692.76 | 145.79 | 86.54 |
| | GP | 4.0 | 7.1 | 45.9 | 3.6 | 24.8 |
| Batch | MSE | 6.31 | 46.61 | 675.35 | 143.05 | 84.54 |
| | GP | 4.0 | 6.8 | 45.3 | 3.5 | 20.4 |
| Proposed | MSE | 6.31 | 46.61 | 675.35 | 143.05 | 84.62 |
| | GP | 4.0 | 6.7 | 45.3 | 3.5 | 21.3 |


[2] S. Kitagami, T. Suganuma, T. Ogino and N. Shiratori, "Proposal of a Multi-agent Based Flexible IoT Edge Computing Architecture Harmonizing Its Control with Cloud Computing," *The Fifth International Symposium on Computing and Networking (CANDAR2017)*, November, 2017.

[3] M. Abdelshkour, "IoT, from Cloud to Fog Computing," http://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing, Mar.2015 (accessed 18 Jun.2017).

[4] P. G. Lopez, A. Montresor, D. Epema, A, Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber and E. Riviere, "Edge-centric Computing: Vision and Challenges," *ACM SIGCOMM Computer Communication Review*, vol.45, issue 5, pp.37-42, Oct.2015.

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

[7] 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.

[8] 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.

[9] H. Miyajima, H. Miyajima and N. Shiratori, "Proposal of Security Preserving Machine Learning of IoT," *Artificial Intelligence Research*, vol.7, no.2, pp.26-33, 2018.

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

[11] H. Miyajima, H. Miyajima and N. Shiratori, "Fast and Secure Edge-computing Algorithms for Classification Problems," *IAENG International Journal of Computer Science*, vol.46, no.4, pp.512-517, 2019.

[12] 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, 2019.

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

[14] M. M. Gupta, L. Jin and N. Homma, "Static and Dynamic Neural Networks," *IEEE Press*, 2003.

[15] 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.

[16] 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.

[17] H. Miyajima, N. Shigei, H. Miyajima, N. Shiratori, "Simplified Security Learning using Vertically Partitioned Data with IoT," *Nonlinear Theory and Its Applications, IEICE*, Vol.12, No.3, pp.412-423, 2021.

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