Fast and Secure Edge-computing Algorithms for Classification Problems

Hirofumi Miyajima, Hiromi Miyajima, and Norio Shiratori

Abstract—**Cloud (computing) system has been widely used in various fields. However, as the number of terminals increases, the limits of the capabilities are also becoming clear. The limits lead to the delay of significant processing time. In order to improve this, the edge (or fog) computing system has been proposed. In the conventional cloud system, users (clients) send all data to the cloud and the cloud returns the computation result to users (clients). On the other hand, in the edge (computing) system, multiple servers called edges are assigned between the cloud and the terminals (or things). In the system, there are two types of servers for cloud and edge. Heavy and normal tasks are processed in the cloud and edge servers, respectively. Then, how is machine learning in the cloud or edge system? The purpose of learning is to find out the relationship (information) lurking in from the collected data. That is, a system with several parameters is assumed and estimated by repeatedly updating the parameters with learning data. Further, there is the problem of security for learning data. How can we build the cloud system to avoid such risk? Secure multiparty computation (SMC) is known as one method realizing secure computation. Many studies on learning methods based on SMC have also been proposed in the cloud system. Then, what kind of learning method is suitable for the edge system based on SMC? In this paper, Neural Gas (NG) algorithms to realize fast and secure processing on edge computing for clustering and classification problems are proposed and the effectiveness of the proposed methods in numerical simulations is shown.**

Index Terms—**IoT, Machine learning, Security, Batch learning, Clustering, Classification problem, Neural Gas.**

I. Introduction

CLOUD (computing) system has been widely used as one technology that supports ICTs. Cloud computing is a system where multiple users (clients) use servers with high capability, so it is possible to reduce operating costs. In conventional cloud computing, data management and calculation processing are collectively performed on servers of the cloud. In IoT (Internet of Things), however, the number of terminals (things) connected to the cloud increases compared with the traditional system. As a result, it is known that the processing capability for each task may be degraded[1], [2]. In order to improve this, the edge (or fog) computing system has been proposed[3], [4]. In the conventional cloud system, users (clients) send all data to the cloud and the cloud returns the computation result to users (clients). In the edge system, multiple servers called edges is connected directly or to close the distance between the cloud and the terminal (or thing). Each server in the edge system does not necessary have a high capacity. But it seems that high processing capability can be realized by efficiently combining multiple servers in the edge system. From the side of terminals, normal tasks can be handled at edges, and the conventional cloud is used for tasks that require large computing power. Then, what kind of paradigm for machine learning with edge computing is needed? The purpose of learning is to find out the relationship (information) lurking in from the collected data. In order to realize this, a system with several parameters is assumed and estimated by repeatedly updating the parameters with learning. Further, users of cloud or edge computing cannot escape the concern about the risk of information leakage. How can we build a cloud or edge computing system to avoid such risks and to perform fast learning? One way to achieve this goal is to use data encryption. Data encryption is an effective way to protect data from risk, but data must be repeatedly encrypted and decrypted each time data processing is done. Therefore, a safe system using distributed processing has attracted attention, and a lot of studies with the cloud have been proposed[5], [6]. SMC is one of the typical model of them[7], [8], [9]. However, there are few studies about the SMC model with IoT. In order to perform it, the effectiveness of batch processing for the BP of the neural network was shown in the previous paper[13]. Further, a fast and secure clustering method for IoT is also proposed[11]. In this paper, NG algorithms to realize fast and secure processing on edge computing for clustering and classification problems are proposed and the effectiveness of the proposed methods is shown in numerical simulations.

Hirofumi Miyajima is Faculty of Informatics, Okayama University of Science, Japan e-mail: miya@mis.ous.ac.jp
Hiromi Miyajima is with Former Kagoshima University.
Norio Shiratori is with Chuo University.

II. Preliminary

A. A configuration of edge computing system

The purpose of the edge system is to perform the effective computation by combining multiple servers with the low capacity to build a system with high processing capability[1], [2], [3]. Fig.1(a) and (b) show two images for the conventional and edge systems, respectively. In the conventional cloud system, each user (client) that needs calculation processing sends all data to the cloud and returns the computation result to user (client) (See Fig.1 (a)). Fig.1(b) is composed of the terminals (or things) directly connected to the cloud and multiple servers (called edges) connected directly or to close the distance between the cloud and the terminal (or things). In order to execute fast computation while maintaining security, how can we share and distribute data among servers? In this paper, a system shown in Fig.2 is assumed as an example for local servers of the edge system.

B. Steepest descent method in machine learning

The purpose of machine learning is to give a method to realize the input/output relation of given learning data by determining the parameters of one system. Since appropriate parameters cannot be found directly, parameters are estimated by adaptively updating the parameters based on SDM (Steepest Descent Method)[12]. Applications of SDM include BP (Back Propagation) learning of neural network, unsupervised learning like k-means and NG (Neural Gas) and fuzzy modeling, etc.

SDM is a way to minimize an evaluation function $T(\boldsymbol{\theta})$ parameterized by a system parameter $\boldsymbol{\theta}\in R^d$ by updating the parameters in the opposite direction of the gradient $\frac{\partial T(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ of the evaluation function to the parameters, where R is the set of all real numbers. The learning rate η determines the size of the steps we take to reach a (local) minimum. The method is performed based on the following equation[12] :

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

That is, the parameter $\boldsymbol{\theta}$ is updated based on Eq.(1) using learning data in order to reach a minimum. There are three methods based on how to use learning data, online, mini-batch and batch. In the following, let us explain the mini-batch method.

Let D be the set of learning data and $Z_i = \{1, \cdots, i\}$ for a positive integer i. The set D is composed of L subsets such as $D = \bigcup_{l=1}^{L} B_l$ and $B_i \cap B_j = \phi$ for $i \neq j \in Z_L$, where $|B_l| = b_l$ for $l \in Z_L$ and $|D| = \sum_{l=1}^{L} b_l$.

Let $t = 1$. Let ε be a small number.

Learning Algorithm A (Mini-batch learning)

Input : The set D of learning data

Output : System parameters $\boldsymbol{\theta}$

Step A1 : The set B_l is given.

Step A2 : The parameter $\boldsymbol{\theta}$ based on Eq.(1) for B_l is updated.

Step A3 : If $\nabla T(\boldsymbol{\theta}) > \varepsilon$, then go to Step 1 with $t \leftarrow t+1$ else algorithm terminates.

If $L = 1$ and $L = |D|$, then the methods are called online and batch ones, respectively.

C. System configuration of secure shared data for SMC

Let us consider conventional works with securely shared data for SMC. In order to solve the problem, three partitioned representation of data such as horizontally, vertically and any partitioned data for SMC are well known[8], [9]. Let us explain about the horizontally partitioned data using an example of Table I. In Table I, a and b are original data (marks) and ID is student identifier. The purpose of computation is to get the average of them.

All the data are shared into two servers, Server 1 and Server 2 as follows:

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

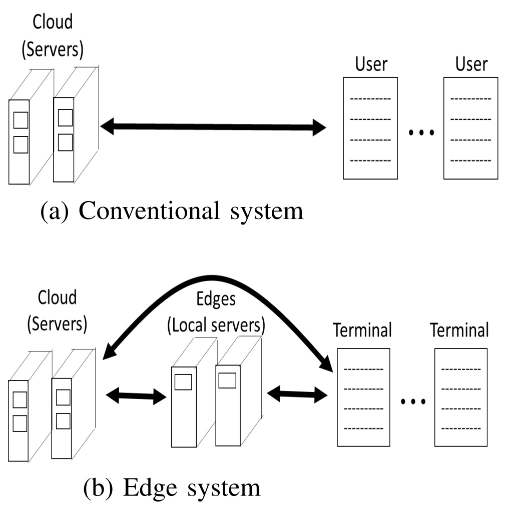
Server 2: dataset for ID=4, 5.

In Server 1, each average for A or B is computed as $(53 + 98 + 36)/3$ and $(46 + 49 + 61)/3$, respectively. In Server 2, each average for A or B is computed as $(56 + 99)/2$ and $(34 + 64)/2$, respectively. As a result, two averages for subsets A and B are 68.4 and 50.8, respectively. Each server cannot know half of the dataset, so security preserving hold.

In the following, secure learning methods are proposed by using horizontally partitioned data.

D. Neural gas method

Vector quantization techniques encode a data space, e.g., a subspace $V \subseteq R^d$, utilizing only a finite set $U = \{\boldsymbol{u}_i | i \in Z_r\}$ of reference vectors (also called cluster centers), where d and r are positive integers.



(a) Conventional system

(b) Edge system

Fig. 1. Cloud and edge systems.

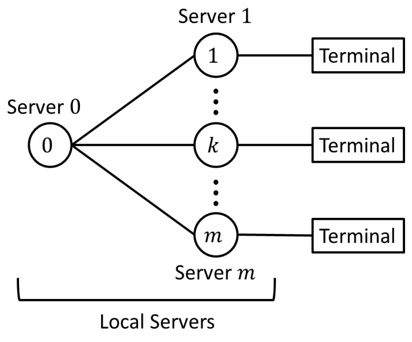


Fig. 2. An example for local servers of edge system.

Let the winner vector $\boldsymbol{u}_{i(\boldsymbol{v})}$ be defined for any vector $\boldsymbol{v} \in V$ as follows:

$$i(\boldsymbol{v}) = \arg\min_{i \in Z_r} ||\boldsymbol{v} - \boldsymbol{u}_i|| \tag{2}$$

From the finite set U, V is partitioned as follows:

$$V_i = \{\boldsymbol{v} \in V | ||\boldsymbol{v} - \boldsymbol{u}_i|| \leq ||\boldsymbol{v} - \boldsymbol{u}_j|| \ for \ j \in Z_r\} \tag{3}$$

The set V and U are called sets of input and reference vectors, respectively.

For NG method[13], the following method is used:

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

$$||\boldsymbol{v} - \boldsymbol{u}_j|| < ||\boldsymbol{v} - \boldsymbol{u}_{i_k}|| \tag{4}$$

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

	ID	Subject A a	Subject B b	
Server 1	1	53	46	Horizontally partitioned data
	2	98	49	
	3	36	61	
Server 2	4	56	34	
	5	99	64	
	average	68.4	50.8	

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

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

$$\triangle\boldsymbol{u}_i = \alpha h_\lambda(k_i(\boldsymbol{v},\boldsymbol{u}_i))(\boldsymbol{v}-\boldsymbol{u}_i) \tag{5}$$

$$h_\lambda(k_i(\boldsymbol{v},\boldsymbol{u}_i)) = \exp\left(-k_i(\boldsymbol{v},\boldsymbol{u}_i)/\lambda)\right) \tag{6}$$

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

where the following function is used as an evaluation one :

$$E = \sum_{\boldsymbol{u}_i\in U}\sum_{\boldsymbol{v}\in V}\frac{h_\lambda(k_i(\boldsymbol{v},\boldsymbol{u}_i))}{\sum_{\boldsymbol{u}_l\in U}h_\lambda(k_l(\boldsymbol{v},\boldsymbol{u}_l))}||\boldsymbol{v}-\boldsymbol{u}_{i(\boldsymbol{v})}||^2 \tag{7}$$

The number λ is called decay constant.

If $\lambda\to 0$, Eq.(5) becomes equivalent to the K-means method[13].

Let $p(\boldsymbol{v})$ be the probability distribution of data vectors for V. Then, NG method is shown as follows[13] :

Learning Algorithm B (Neural Gas)

Input : The set V of input vectors.

Output : The set U of reference vectors.

Step B1 : The initial values of reference vectors are set randomly. The learning coefficients α_{int} and α_{fin} are set. Let T_{max} be the maximum number of learning time.

Step B2 : Let $t = 1$.

Step B3 : Give a data $\boldsymbol{v}\in V$ based on $p(\boldsymbol{v})$ and each neighborhood-ranking $k_i(\boldsymbol{v},\boldsymbol{u}_i)$ is determined for $i\in Z_r$.

Step B4 : Each reference vector $\boldsymbol{u}_i$ for $i\in Z_r$ is updated based on Eq.(5)

Step B5 : If $t\geq T_{max}$, then the algorithm terminates and the set $U = \{\boldsymbol{u}_i|i\in Z_r\}$ of reference vectors for $\boldsymbol{V}$ is obtained else go to Step B3 as $t\leftarrow t+1$.

E. Adaptively local linear mapping

The aim in this section is to adaptively approximate the function $y = f(\boldsymbol{v})$ with $\boldsymbol{v}\in V\subseteq R^d$ and $y\in R$ using NG[13]. That is, a supervised learning using NG is introduced. The set V denotes the function's domain. Let us consider n computational units, each containing a reference vector $\boldsymbol{u}_i$ together with a constant a_{i0} and d-dimensional vectors $\boldsymbol{a}_i$. Learning Algorithm B assigns each unit i to a subregion V_i as defined in Eq.(3), and the coefficients a_{i0} and $\boldsymbol{a}_i$ define a linear mapping

$$g(\boldsymbol{v}) = a_{i0} + \boldsymbol{a}_i(\boldsymbol{v}-\boldsymbol{u}_i) \tag{8}$$

from R^d to R over each of the Voronoi diagram V_i(See Fig.3). Hence, the function $y = f(\boldsymbol{v})$ is approximated by $\tilde{y} = g(\boldsymbol{v})$ with

$$g(\boldsymbol{v}) = a_{i(\boldsymbol{v})0} + \boldsymbol{a}_{i(\boldsymbol{v})}(\boldsymbol{v}-\boldsymbol{u}_{i(\boldsymbol{v})}) \tag{9}$$

where $i(\boldsymbol{v})$ denotes unit i with its $\boldsymbol{u}_i$ closest to $\boldsymbol{v}$.

To learn the input-output mapping, a series of training steps by giving the set of learning data $D = \{(\boldsymbol{x}^p, y_p)|p\in Z_P\}$ is performed. In order to obtain the output coefficients a_{i0} and $\boldsymbol{a}_i$, the mean squared error $\sum_{\boldsymbol{v}\in V}(f(\boldsymbol{v})-g(\boldsymbol{v}))^2$ between the actual and the obtained output, averaged over subregion $\boldsymbol{V}_i$, to be minimal is required for each i. SDM with respect to a_{i0} and $\boldsymbol{a}_i$ yields[13]:

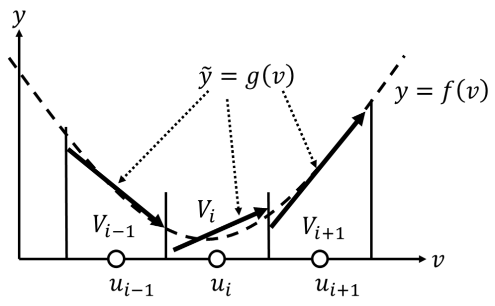


Fig. 3. The concept of local linear mapping : The set V_i is one composed of element $\boldsymbol{v}$ closest to the reference vector u_i. The interval V_i is approximated by a local linear mapping.

$$\triangle a_{i0} = \alpha' h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{u}_i))(y - a_{i0} - \boldsymbol{a}_i(\boldsymbol{v}-\boldsymbol{u}_i)) \tag{10}$$

$$\triangle\boldsymbol{a}_i = \alpha' h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{u}_i))(y - a_{i0} - \boldsymbol{a}_i(\boldsymbol{v}-\boldsymbol{u}_i))(\boldsymbol{v}-\boldsymbol{u}_i) \tag{11}$$

where $\alpha' > 0$ and $\lambda' > 0$.

Remark that the case of k-means is included as a special one.

The algorithm is introduced as follows[13] :

Learning Algorithm C (Adaptively approximation using local linear mapping by NG)

Input : Learning data $D = \{(\boldsymbol{x}^p, y_p)|p\in Z_P\}$ and $D^* = \{\boldsymbol{x}^p|p\in Z_P\}$. The probability distribution $p(\boldsymbol{v})$.

Output : The set U of reference vectors and the coefficients a_{i0} and $\boldsymbol{a}_i$ for $i\in Z_r$.

Step C1 : The set U of reference vectors is determined using D^* by Algorithm B. The subregion V_i for $i\in Z_r$ is determined using U, where V_i is defined by Eq.(3), $V^d = \cup_{i=1}^r V_i$ and $V_i\cap V_j = \emptyset(i\neq j)$. Let T_{max} be the maximum number of learning time.

Step C2 : Parameters a_{i0} and $\boldsymbol{a}_i$ for $i\in Z_r$ are set randomly. Let $t = 1$.

Step C3 : A learning data $(\boldsymbol{x}, y)\in D$ is selected based on $p(\boldsymbol{x})$. The rank $k_i(\boldsymbol{x},\boldsymbol{u}_i)$ of $\boldsymbol{x}$ for the set V_i is determined.

Step C4 : Parameters a_{i0} and $\boldsymbol{a}_i$ for $i\in Z_r$ are updated based on Eqs.(10) and (11).

Step C5 : If $t\geq T_{max}$, then the algorithm terminates else go to Step C3 with $t\leftarrow t+1$.

Fig.3 shows the figure to explain the local linear mapping.

III. Proposed methods of clustering and classification problems for edge computing

In this chapter, learning methods of edge computing for clustering and classification problems are proposed. The former and the latter are unsupervised and supervised learning, respectively. In order to realize this, algorithms B and C in Chapter II are used. In chapters III.A and III.B, online and batch learning methods for edge computing for clustering problems are proposed. In chapter III.C, a learning method for classification problems for edge computing is proposed. The system in Fig.2 composed of Server 0 and m servers is used.

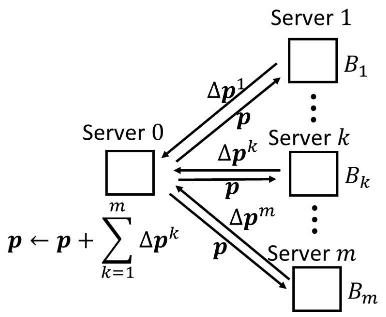


Fig. 4. The figure of explanation for the proposed algorithm : Based on the horizontally partitioned data, the set B_k is given in advance to Server k. The update amount $\triangle \boldsymbol{p}^k$ for the set B_k of parameter $\boldsymbol{p}$ in Server k is computed and sent to Server 0. In Server 0, the parameter $\boldsymbol{p}$ is updated and sent to each Server.

A. Online unsupervised learning for edge computing

In order to compare the ability of the online method with the batch method, an online learning method will be introduced. First, the set D of learning data is divided into m pieces of subsets and each subset B_k is given to Server k, where $D = \cup_{k=1}^{m} B_k$. The set W of reference vectors is randomly selected and sent to each Server. In Step 1, a constant number $k^* \in Z_m$ is selected randomly and is sent to each Server. In Step 2, the task at Server k^* is performed. That is, an element $\boldsymbol{x} \in B_{k^*}$ is selected randomly, $\triangle \boldsymbol{w}_i^{k^*}$ is computed based on Eq.(6) and sent to Server 0. In Step 3, each element of the set W is updated using $\triangle \boldsymbol{w}_i^{k^*}$ and the set W is sent to each Server. In Step 4, the set W of each Server is renewed. In Step 5, check if the maximum number of learning is performed. The final result of the set W is obtained in Server 0 and each Server. The algorithm is shown in Table II.

B. Batch unsupervised learning for edge computing

A batch learning method will be proposed. As the case of online learning, the set D of learning data is divided into m pieces of subsets and each subset B_k is given to the Server k. The set W of reference vectors is randomly selected and sent to each Server. In Step 1, the update amount $\triangle \boldsymbol{w}_i^k$ for each element of W in Server k is computed by adding with $\boldsymbol{x} \in B_k$ and the result is sent to Server 0. In Step 2, all update amounts sent from each server are added with $k \in Z_m$ and each element $\boldsymbol{w}_i$ of W is updated. The set W is sent to each Server. In Step 3, the set W of each Server is renewed. In Step 4, check if the maximum number of learning is performed. The final result of the set W is obtained in Server 0. The algorithm is shown in Table III. The idea of batch learning for the proposed method is shown in Fig.4.

C. Batch learning using local linear mapping for edge computing

Let D be the set of learning data. By using the set $D^* = \{\boldsymbol{x}^p | p \in Z_P\}$, the set W of reference vectors is approximated using Leaning Algorithm B. In each server, the initial parameters of linear mapping for each Voronoi region defined by the sets D and W are set randomly. A batch learning using local linear mapping is proposed as shown in Table IV. In Step 1, the rank $k_i(\boldsymbol{x}, \boldsymbol{w}_i)$ of input $\boldsymbol{x}$ and vector $\boldsymbol{w}_i$ for set W is calculated in each Server. By using the result, update amounts $\triangle a_{i0}^k$ and $\triangle \boldsymbol{a}_i^k$ for the constant a_{i0}^k and the coefficient $\boldsymbol{a}_i^k$ of local linear functions are computed, respectively, and they are added with $\boldsymbol{x} \in B_k$. In Step 2, update amounts $\triangle a_{i0}$ and $\triangle \boldsymbol{a}_i$ are computed by adding with $k \in Z_m$ and parameters a_{i0} and $\boldsymbol{a}_i$ for $i \in Z_r$ are updated, respectively. The results are sent to each Server. In Step 3, the learning error $E_k(t)$ for B_k is computed by using updated local linear functions and the result is sent to Server 0. In Step 4, the learning error $E(t)$ for the set D is computed by adding $E_k(t)$ with $k \in Z_m$. Further, if $E(t)$ is smaller than the threshold θ or t is larger than the maximum number T_{max} of learning time, then the algorithm terminates else go to Step 1 with $t \leftarrow t+1$. Likewise, an online learning method is proposed.

IV. Numerical Simulations

A. Clustering problems

Four real-world datasets from UCI machine learning repository have been considered as shown in Table V[14], where #data, #input, and #class mean the numbers of data, input variables, and classes, respectively. As the initial condition, the initial values of W are selected randomly from $[0, 1]$ and the maximum number of learning times are $100 \times \#$data. Let $\varepsilon_{int} = 0.1$ and $\varepsilon_{fin} = 0.01$. The problem is how each dataset is approximately by reference vectors. Tables VI and VII show the results of the misclassification rates. The misclassification rate means the ratio of the misclassified data to all data. Each result is the average value from twenty trials. Conventional online, batch, Proposed online and batch mean the conventional online, batch learning methods, the proposed online and batch learning methods, respectively. In this simulation, two scenarios are performed : the first one is $r = \#$class that each class is approximated by one reference vector and the second one is $r = 4 \times \#$class that each class is approximated by four reference vectors.

Both results show that approximation accuracy for conventional and the proposed methods are almost the same. Therefore, the proposed method has good accuracy and security preserving. Since it is difficult to evaluate direct computing time, the time required for parameter updating of one time for learning data is estimated. In online processing, let $O_1(1)$ and $O_2(1)$ be the computing time of the error and the time to update parameters for each data of the server, respectively. Then, the time complexity of the case is $P(O_1(1) + O_2(1))$, where P is the number of data. That is, the total time complexity is $O(P)$. On the other hand, in batch processing, when the computing time of parameter updating for (P/m) pieces of learning data is assumed to be $O_3(P/m)$, the computing time for all learning data is $O(1 \cdot O_3(P/m))$, because all computing for edge server is done at a time. The time complexity is $O(O_4(m) + 1 \cdot O_3(P/m))$, assuming that the computing time required for updating at Server 0 is $O_4(m)$ by using the update amount of each Server. In this case, since the first term is smaller than the second term, $O(O_4(m) + 1 \cdot O_3(P/m)) \approx O(1 \cdot O_3(P/m))$ is estimated. As a result, the speedup of m times in batch processing is expected.

B. Classification problems

In these simulations, 5-fold cross validation as an evaluation method is used: In 5-fold cross-validation, all data

TABLE II
AN ONLINE UNSUPERVISED LEARNING FOR EDGE COMPUTING OF SMC

	Server 0	Server k
Initial condition	The set W of reference vectors is selected randomly and sent to each Server, where $W=\{\boldsymbol{w}_i \mid i\in Z_r\}$. α, T_{max} and θ are given. Set $t=1$.	The subset B_k of learning data where $D=\cup_{k=1}^{m} B_k$.
Step 1	Select a server number k^* and send it to each Server.	
Step 2		If $k=k^*$, then select a data $\boldsymbol{x}\in B_k$ and calculate $\triangle\boldsymbol{w}_i^{k^*}=\varepsilon\cdot h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}_i))\cdot(\boldsymbol{x}-\boldsymbol{w}_i)$ for $i\in Z_r$, where $h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}_i))=\exp(-k_i(\boldsymbol{x},\boldsymbol{w}_i)/\lambda)$ and send them to Server 0.
Step 3	Calculate $\boldsymbol{w}_i\leftarrow\boldsymbol{w}_i+\alpha\triangle\boldsymbol{w}_i^{k^*}$ for $i\in Z_r$ and send them to each Server.	
Step 4		Update the set W.
Step 5	If $t\geq T_{max}$ then the algorithm terminates else go to Step 1 with $t\leftarrow t+1$.	

TABLE III
A BATCH UNSUPERVISED LEARNING FOR EDGE COMPUTING OF SMC

	Server 0	Server k
Initial condition	α, T_{max} and θ are given. Set $t=1$.	The subset B_k of learning data where $D=\cup_{k=1}^{m} B_k$
Step 1		Calculate $\triangle\boldsymbol{w}_i^k=\sum_{\boldsymbol{x}\in B_k}\varepsilon\cdot h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}_i))\cdot(\boldsymbol{x}-\boldsymbol{w}_i)$ for $i\in Z_m$, where $h_\lambda(k_i(\boldsymbol{x},\boldsymbol{w}_i))=\exp(-k_i(\boldsymbol{x},\boldsymbol{w}_i)/\lambda)$ for $i\in Z_r$ and send them to Server 0.
Step 2	Calculate $\boldsymbol{w}_i\leftarrow\boldsymbol{w}_i+\alpha\sum_{k=1}^{m}\triangle\boldsymbol{w}_i^k$ for $i\in Z_r$ and send them to each Server.	
Step 3		Update the weight W.
Step 4	If $t\geq T_{max}$ then the algorithm terminates else go to Step 1 with $t\leftarrow t+1$.	

TABLE IV
LEARNING METHOD USING LOCAL LINEAR MAPPING

	Server 0	Server k
Initial condition	The set W is determined by the unsupervised learning for Edge computing and sent to each Server, where $W=\{\boldsymbol{w}_i \mid i\in Z_r\}$. α, T_{max} and θ are given. Set $t\leftarrow 1$.	The set B_k is given. Parameters a_{i0}^k and $\boldsymbol{a}_i^k$ are set randomly. Let $t\leftarrow 1$.
Step 1		Calculate $e_i(\boldsymbol{x},\boldsymbol{w}_i)$ for $\boldsymbol{x}\in B_k$ and $i\in Z_r$. Calculate $\triangle a_{i0}^k=\sum_{\boldsymbol{x}\in B_k}\alpha' h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{u}_i))(y-a_{i0}-\boldsymbol{a}_i(\boldsymbol{v}-\boldsymbol{u}_i))$ $\triangle\boldsymbol{a}_i^k=\sum_{\boldsymbol{x}\in B_k}\alpha' h_{\lambda'}(k_i(\boldsymbol{v},\boldsymbol{u}_i))(y-a_{i0}-\boldsymbol{a}_i(\boldsymbol{v}-\boldsymbol{u}_i))(\boldsymbol{v}-\boldsymbol{u}_i)$ where $\alpha'>0$ and $\lambda'>0$ and send to Server 0.
Step 2	Calculate $\triangle a_{i0}\leftarrow a_{i0}+\alpha\sum_{k=1}^{m}\triangle a_{i0}^k$ $\boldsymbol{a}_i\leftarrow\boldsymbol{a}_i+\alpha\sum_{k=1}^{m}\triangle\boldsymbol{a}_i^k$ for $i\in Z_r$ and send them to each Server.	
Step 3		Calculate $E_k(t)=\sum_{\boldsymbol{x}\in B_k}(g(\boldsymbol{x})-d(\boldsymbol{x}))^2$ and send them to Server 0.
Step 4	Compute $E(t)=\frac{1}{\lvert D\rvert}\sum_{k=1}^{m}E_k(t)$. If $E<\theta$ or $t\geq T_{max}$, then the algorithm terminates else go to Step 1 with $t\leftarrow t+1$.	

TABLE V
THE DATASET FOR PATTERN CLASSIFICATION

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

TABLE VI
CONVENTIONAL AND PROPOSED K-MEANS

		Iris	Wine	Sonar	BCW
Online ($r=$ #classes)	Conventional	9.9	8.4	44.9	3.9
	Proposed	9.9	9.6	45.0	3.9
Batch ($r=$ #classes)	Conventional	8.4	6.2	45.7	3.9
	Proposed	5.5	6.6	45.6	3.9
Online ($r=4\times$#classes)	Conventional	4.1	7.9	36.1	3.0
	Proposed	4.1	7.0	36.3	3.0
Batch ($r=4\times$#classes)	Conventional	3.8	6.6	35.6	2.9
	Proposed	3.7	6.7	35.5	2.9

are randomly partitioned into 5 equal size subsets. Of the 5 subsets, a single subset is kept as data for testing the model, and the remaining 4 subsets are used as training data. The cross-validation process is repeated 5 times (the folds) with each of 5 subsets used exactly once as the validation (test) data. The five results from the folds can then be averaged to produce a single estimation. In this simulation, two scenarios

TABLE VII
CONVENTIONAL AND PROPOSED NG

		Iris	Wine	Sonar	BCW
Online ($r = \#$classes)	Conventional	4.0	6.9	45.1	3.5
	Proposed	4.0	7.2	44.9	3.6
Batch ($r = \#$classes)	Conventional	4.0	6.6	45.2	3.5
	Proposed	4.0	6.6	45.2	3.5
Online ($r = 4\times\#$classes)	Conventional	3.6	5.3	34.0	2.9
	Proposed	4.1	5.8	33.9	2.9
Batch ($r = 4\times\#$classes)	Conventional	4.1	7.0	35.9	3.1
	Proposed	3.8	6.4	35.9	2.9

TABLE VIII
CONVENTIONAL AND PROPOSED K-MEANS (LLM) ($r = \#$CLASSES)

		Iris	Wine	Sonar	BCW
Conventional online	Training	3.7	2.5	0.2	2.8
	Test	4.0	5.1	25.0	3.7
Proposed online	Training	3.6	2.3	0.3	2.8
	Test	3.9	5.3	25.1	3.5
Conventional batch	Training	3.6	2.5	1.3	2.9
	Test	3.8	4.8	24.6	3.6
Proposed batch	Training	3.3	2.4	2.1	2.9
	Test	3.6	4.5	26.7	3.6

are performed : the first one is $r = \#$class that each class is approximated by one reference vector and the second one is $r = 4\times\#$class that each class is approximated by four reference vectors.

Tables VIII, IX, X and XI show the results of the comparison between the conventional and the proposed methods. In each box of Tables VIII, IX, X and XI, Training and Test mean the rate (%) of misclassified data for training and test data, respectively. Each value is average from five trials. Further, LLM means local linear mapping.

Both results show that approximation accuracy for conventional and the proposed methods is almost the same. Therefore, the proposed method also has good accuracy in the cases.

V. CONCLUSION

In this paper, secure and fast learning methods for clustering and classification problems of IoT were proposed and its effectiveness was shown by numerical simulations. In Section 2, cloud and edge computing systems, a secure data sharing mechanism used in this paper were explained. Further, the conventional NG and adaptively local linear mapping were introduced. In Section 3, learning methods suitable for edge computing for clustering and classification problems were proposed. In section 4, numerical simulations were performed to show the performance of the proposed methods. The idea of the proposed methods is to combine parallelism of batch learning for SDM and the horizontally partitioned data for SMC. This idea seems applicable to other learning problems. Although the superiority of batch learning was shown by analysis of time complexity, verification in implementation is future work

TABLE IX
CONVENTIONAL AND PROPOSED K-MEANS (LLM) ($r = 4\times\#$CLASSES)

		Iris	Wine	Sonar	BCW
Conventional online	Training	3.9	2.2	0.3	2.3
	Test	4.4	8.6	20.2	3.3
Proposed online	Training	4.1	2.3	0.3	2.3
	Test	4.4	9.1	19.9	3.3
Conventional batch	Training	4.0	2.2	0.3	2.3
	Test	4.7	8.9	19.9	3.1
Proposed batch	Training	4.0	2.2	0.8	2.3
	Test	4.4	8.9	19.8	3.3

TABLE X
CONVENTIONAL AND PROPOSED NG (LLM) ($r = \#$CLASSES)

		Iris	Wine	Sonar	BCW
Conventional online	Training	4.0	2.5	0.2	3.0
	Test	4.0	4.6	25.5	3.5
Proposed online	Training	3.9	2.6	0,3	2.9
	Test	4.0	4.6	25.4	3.5
Conventional batch	Training	4.0	2.4	0.4	2.9
	Test	4.0	4.6	25.2	3.4
Proposed batch	Training	3.9	2.4	4.3	2.8
	Test	3.9	4.6	24.9	3.1

TABLE XI
CONVENTIONAL AND PROPOSED NG (LLM) ($r = 4\times\#$CLASSES)

		Iris	Wine	Sonar	BCW
Conventional online	Training	3.9	2.2	0.3	2.3
	Test	4.8	9.3	19.5	3.4
Proposed online	Training	4.0	2.3	0.3	2.3
	Test	4.4	9.1	19.5	3.4
Conventional batch	Training	4.1	2.1	0.3	2.3
	Test	4.8	8.7	19.5	3.4
Proposed batch	Training	4.0	2.1	0.1	2.3
	Test	4.3	8.6	19.9	3.5

REFERENCES

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

[2] 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] A. Shamir, "How to share a secret", Communications of the ACM, Vol.22, Issue 11, pp.612-613, 1979.

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

[7] S. Subashini, et al., "A survey on security issues in service delivery models of cloud computing", J. Network and Computer Applications, Vol.34,pp.1-11, 2011.

[8] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami and N. Shiratori, "New Privacy Preserving Back Propagation Learning for Secure Multiparty Computation", Journal of Computer Science, Vol.43, No.3, pp.270-276, 2016.

[9] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami and N. Shiratori, "New Privacy Preserving Clustering Methods for Secure Multiparty Computation", Artificial Intelligence Research, Vol.6, No.1, pp.27-36, 2017.

[10] 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.

[11] H. Miyajima, H. Miyajima and N. Shiratori, "Proposal of Fast and Secure Clustering Method for IoT", Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2019, 13-15 March, 2019, Hong Kong, pp.1-6.

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

[13] T. M. Martinetz, S. G. Berkovich and K. J. Schulten, Neural Gas Network for Vector Quantization and its Application to Time-series Prediction, IEEE Trans. Neural Network, 4, 4, pp.558-569, 1993.

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