A Proposal of Back Propagation Learning for Secure Multi-Party Computation Methods

Hirofumi Miyajima†1, Noritaka Shigei†2, Hiromi Miyajima†3,
Yohtaro Miyanishi†4, Shinji Kitagami†5 and Norio Shiratori†6

***Abstract*—Many studies have been done with the security of cloud computing. Data encryption is one of typical approaches. However, complex computing requirement needs a great deal of time and effort for the system. Therefore, studies on secure sharing and computing methods are desired to avoid secure risks being abused or leaked. The secure multi-party computation (SMC) is one of these methods. So far, some studies have been done with SMC. Specifically, SMC with secure shared data in addition and multiplication forms is proposed and applied to arithmetic operation and simple statistical computation. However, complex calculation processing such as BP (Back Propagation) learning has never proposed yet. In this paper, we propose a learning method for SMC on cloud computing system and prove the validity of it. Further, the performance of the proposed method is shown in numerical simulations.**

***Index Terms*—cloud computing, secure multi-party computation, BP, Perceptron-type learning.**

I. Introduction

Cloud computing is service to provide computing resources through the Internet for the unspecified number of individual or a company [1], [2]. The client can realize reduction of the operational cost and the increase and decrease of resources flexibly without having one's resources. In addition, the spreading of computing of cloud computing allows use such as big data analysis to analyze enormous information accumulated by the client, and to create market value of data [1]. On the other hand, the client of cloud computing cannot escape from anxiety about the possibility of information being abused or leaked. How can we construct a cloud computing system to avoid the above risk? For this purpose, data encryption has been one of typical approaches [3], [4]. Data encryption is an effective method to protect data from external risk, but the encrypted data must be decrypted to get plain texts when processed in a cloud. Therefore, safe system for distributed processing with secure data attracts attention, and a lot of studies have been done [5], [6]. A method to realize secure computation is secure multi-party method [7]–[11], which is currently applied to certain applications, but for limited function calculations. Further, Miyanishi proposed a simple method to share data and applied to statistical computation [12]. However, complex calculation processing such as BP learning has never proposed yet.

In this paper, we propose a learning method for SMC on cloud computing system and prove the validity of it. Further, the performance of the proposed method is shown in numerical simulations. The aim is to realize the learning using secure sharing and computing for data being on cloud system. In Section 2, we explain cloud computing system and how to share the data. Further, Perceptron-type and BP learning are introduced. Though Perceptron-type learning is a special case of BP learning, it is introduce to help understanding the proposed algorithm for BP learning. In Section 3, we propose perceptron and BP learning on cloud computing system and show the validity of the algorithm. Further, numerical simulations are performed to show the performance of the proposed method.

Affiliation: Graduate School of Science and Engineering, Kagoshima University, 1-21-40 Korimoto, Kagoshima 890-0065, Japan
corresponding author to provide email: miya@eee.kagoshima-u-ac.jp
†1 email: k3768085@kadai.jp
†2 email: shigei@eee.kagoshima-u-ac.jp
†3 email: miya@eee.kagoshima-u-ac.jp
Affiliation: Information Systems Engineering and Management, Inc., Tokyo, Japan
†4 email: miyanisi@jade.dti.ne.jp
Affiliation: Waseda University, Tokyo, Japan
†5 email: kitagami.shinji@meltec.co.jp
Affiliation: Waseda University, Tokyo, Japan
†6 email: norio@shiratori.riec.tohoku.ac.jp

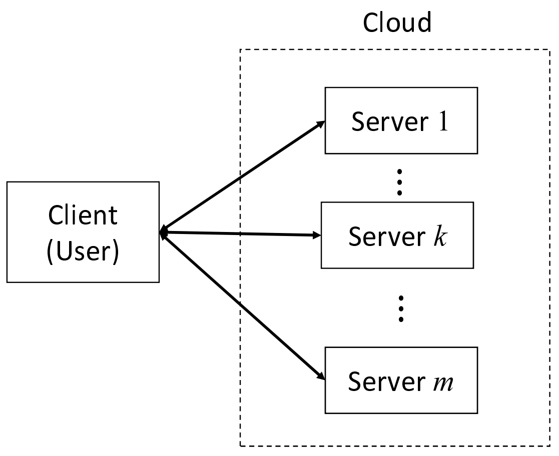

Fig. 1. A configuration of cloud computing system.

II. Preliminary

A. System configuration of cloud system

Fig.1 shows a system of cloud computing used in this paper. The system is composed of a client and cloud with m servers. The client sends data to each server and each server memorize them. If the client requires data processing, each server performs one's computation and sends each result to the client. The client computes the final result using them. If the result is not obtained by one processing, data processing between the client and cloud is iterated until the final result is obtained. The problem is how data are shared and the computation for each server is carried out.

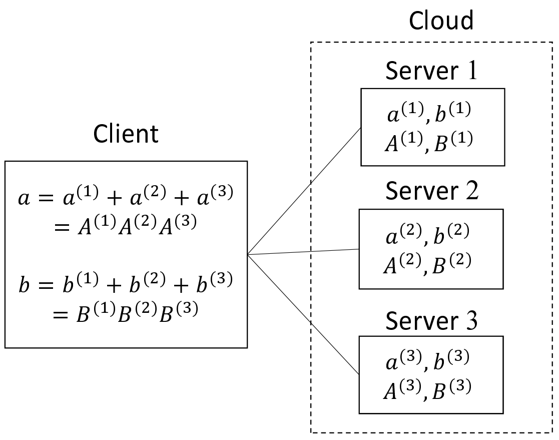

Fig. 2. The representation of secure shared data.

B. The representation of secure shared data

Let us explain data representation for secure multi-party computation used in this paper [10], [12]. Let a and b be two positive integers and $m = 3$ for the number of servers. First, two integers a and b are shared into three real numbers. Let $a = a^{(1)} + a^{(2)} + a^{(3)}$ and $b = b^{(1)} + b^{(2)} + b^{(3)}$ as addition form and $a = A^{(1)}A^{(2)}A^{(3)}$ and $b = B^{(1)}B^{(2)}B^{(3)}$ as the multiplication form (See Fig.2). Then the following results hold:
1)$a + b = (a^{(1)} + b^{(1)}) + (a^{(2)} + b^{(2)}) + (a^{(3)} + b^{(3)})$
2)$a - b = (a^{(1)} - b^{(1)}) + (a^{(2)} - b^{(2)}) + (a^{(3)} - b^{(3)})$
3)$ab = (A^{(1)}B^{(1)})(A^{(2)}B^{(2)})(A^{(3)}B^{(3)})$
4)$a/b = (A^{(1)}/B^{(1)})(A^{(2)}/B^{(2)})(A^{(3)}/B^{(3)})$

That is, four basic operations of arithmetic (addition, subtraction, multiplication, and division) hold as integration of sub computation by each server. In this case, each server can not know the original data a and b. Further, let us explain how to compute the sum and average for results processing using shared data in addition form.

In Table I, a and b are original data (marks) and ID is student's identifier. Let us show a calculation example of shared data as follow [12]:
$a = a^{(1)} + a^{(2)}$: $a^{(1)} = a(r_1/10)$ and $a^{(2)} = a(1 - r_1/10)$
$b = b^{(1)} + b^{(2)}$: $b^{(1)} = b(r_1/10)$ and $b^{(2)} = b(1 - r_2/10)$
$a = A^{(1)}A^{(2)}$: $A^{(1)} = \sqrt{a}(r_1/10)$ and $A^{(2)} = \sqrt{a}(10/r_1)$
$b = B^{(1)}B^{(2)}$: $B^{(1)} = \sqrt{b}(r_2/10)$ and $B^{(2)} = \sqrt{b}(10/r_2)$

Where r_1 and r_2 are real random numbers for $-9 \leq r_1 \leq 9$ and $0.2 \leq r_2 \leq 9$, $r_1 \neq 1$, $r_2 \neq 1$, respectively. For example, $a^{(1)}$ and $a^{(2)}$ for ID=1 are computed as $a^{(1)} = 50 \times (4/10) = 20$ and $a^{(2)} = 50 \times (1 - 4/10) = 30$ and data $A^{(1)}$ and $A^{(2)}$ for ID=1 are computed as $A^{(1)} = \sqrt{50} \times (9/10) = 6.314$ and $A^{(2)} = \sqrt{50} \times (10/9) = 7.856$, respectively. Note that Server 1 has all the data in column-wise of $a^{(1)}$, $b^{(1)}$, $A^{(1)}$ and $B^{(1)}$ for each ID and Server 2 has all the data in column-wise of $a^{(2)}$, $b^{(2)}$, $A^{(2)}$ and $B^{(2)}$ for each ID as shown in Fig.2.

Let us explain how to compute the sum and average for subject A using data a. Server 1 and Server 2 compute each sum of $a^{(1)}$ and $a^{(2)}$. In this case, each sum in column-wise for $a^{(1)}$ and $a^{(2)}$ is -23 and 328, respectively. As a result, the total sum 305 is obtained from 328-23. Likewise, the average 61 is obtained from 65.6-4.6.

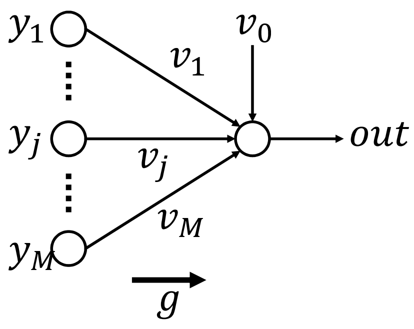

Fig. 3. A neural element.

In the following, we propose a learning method using secure shared data in multiplication form.

C. Perceptron-type learning for a neural element

Perceptron-type learning is a method to adjust the weight parameters to match the output of the neuron and the desired output. Desired output for input are called learning data. In Perceptron-type learning, the weight parameters are updated repeatedly using the difference between the output of the neuron and the desired output until the difference becomes sufficiently small. It is known that the results always converge if the linearly separable condition is satisfied [14].

Let us explain the conventional neuron [14]. Let $Z_i = \{1, - - -, i\}$ for a positive integer i. The output y for each neuron is given by

$$u = \sum_{j=0}^{M} v_j y_j \tag{1}$$
$$out = g(u) \tag{2}$$

where y_j is the j-th input, u is the internal potential of the neuron, $g(u)$ is the output function, v_j is the weight for the j-th input, v_0 is the threshold and $y_0 = 1$ (See Fig. 3).

Let the evaluation function for input data $\boldsymbol{y}$ be defined as follows:

$$E = \frac{1}{2}\left(g(\boldsymbol{y}) - d(\boldsymbol{y})\right)^2, \tag{3}$$

where $d(\boldsymbol{y})$ is the desired output for $\boldsymbol{y}$.

In order to minimize the function E, the weights for the neuron are updated as follows:

$$\triangle v_j = \frac{\partial E}{\partial v_j} = (g(\boldsymbol{y}) - d(\boldsymbol{y}))\frac{\partial g(u)}{\partial u} y_j \tag{4}$$
$$v_j(t+1) = v_j(t) + K_{\triangle} v_j(t), \tag{5}$$

where K is a learning constant and $j \in Z_M$

In order to consider the simple case, let us assume that $\frac{\partial g(u)}{\partial u} = 1$. As a result, learning formula for Perceptron-type learning is shown as follows:

$$v_j(t+1) = v_j(t) + K\left(g(\boldsymbol{y}) - d(\boldsymbol{y})\right) y_j, \tag{6}$$

where t is the number of learning time.

Then, learning algorithm for learning data $\{(\boldsymbol{y}^l, d(\boldsymbol{y}^l)) | l \in Z_L\}$ is shown as follows [14]:

TABLE I
DATA ON SERVER 1 AND SERVER 2.

ID	subject A	subject B	Additional form					Multiplication 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	6.314	7.856	8.050	9.938
2	40	50	-6	-24	64	-30	80	2	1.265	31.623	1.414	35.355
3	65	30	2	13	52	6	24	0.8	0.645	100.778	0.438	68.465
4	70	62	-8	-56	126	-49.6	111.6	5	4.183	16.733	3.937	15.748
5	80	40	3	24	56	12	28	4	3.578	22.361	2.530	15.811
sum	305	262		-23	328	-29.6	291.6					
average	61	52.4		-4.6	65.6	-5.92	58.32					

Step1: Given the maximum number of learning t_{max}. Let $t = 1$. The initial assignment of $v_j (j \in Z_M)$ is set randomly.
Step 2: Let $l = 1$.
Step 3: Select a data $(\boldsymbol{y}^l, d(\boldsymbol{y}^l))$.
Step 4: Compute the output $g(\boldsymbol{y}^l)$ for input $\boldsymbol{y}^l$.
Step 5: Compute the error $\left(g(\boldsymbol{y}^l) - d(\boldsymbol{y}^l)\right)$.
Step 6: Compute the updated value $\triangle v_j(t)$ using Eq.(4).
Step 7: Update the weight as $v_j(t+1) = v_j(t) + K\triangle v_j(t)$.
Step 8: If $l \neq L$, then go to Step 3 with $l \leftarrow l+1$ and $t \leftarrow t+1$ else go to Step 9.
Step 9: If $t \neq T_{max}$, then go to Step 2 with $t \leftarrow t+1$ else the algorithm terminates.

D. BP method for neural networks

Let us consider the case of n input and one output without loss of generality. Let $\boldsymbol{z}^l \in J^N$ for $l \in Z_L$ and $d : J^N \rightarrow J$, where $J = [0, 1]$. Giving learning data $\{(\boldsymbol{z}^l, d(\boldsymbol{z}^l)) | l \in Z_L\}$, let us determine the three-layered neural network identifying the learning data by BP method [13], [14]. Let $h = g \circ e : J^N \rightarrow J$ be the function defined by a neural network. Let the number of elements in second layer be M. Let $\boldsymbol{w}_j$ for $j \in Z_M$ and $\boldsymbol{v}$ be weights for the second and output layers, respectively. Then g and e are defined as follows (See Fig.4):

$$\begin{aligned} y_j &= e_j(\boldsymbol{z}) = \tau\left(\sum_{i=0}^{N} w_{ij} z_i\right), \\ z_0 &= 1, \\ \tau(u) &= \frac{1}{1+\exp(-u)} \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{z} &= (z_1, \cdots, z_N) \in J^N \\ \boldsymbol{y} &= (y_1, \cdots, y_M) \in J^M \end{aligned}$$

and w_{0j} means the threshold value.

Further,

$$\begin{aligned} g(\boldsymbol{y}) &= \tau\left(\sum_{j=0}^{M} v_j y_j\right), \\ y_0 &= 1, \end{aligned}$$

where v_0 means the threshold value.

Then, the evaluation function is defined as follow:

$$E = \frac{1}{2L} \sum_{l=1}^{L} \left(h(\boldsymbol{z}^l) - d(\boldsymbol{z}^l)\right)^2 \tag{7}$$

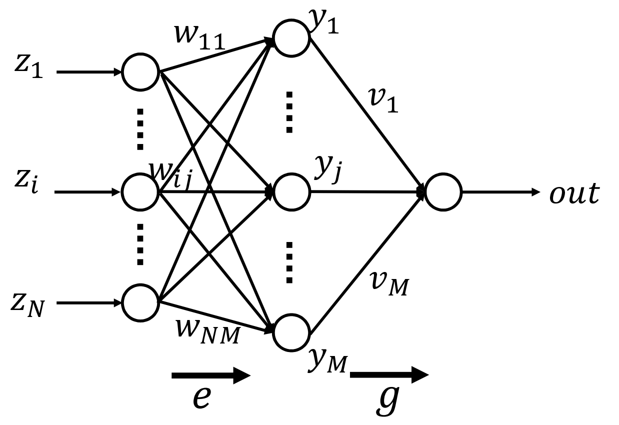

Fig. 4. Three-layered neural network

The weights $\boldsymbol{w}$ and $\boldsymbol{v}$ are updated based on BP method as follow [14]:

$$\triangle v_j = -\alpha_1 \delta_{1j}(\boldsymbol{z}^l) e_j(\boldsymbol{z}^l) \tag{8}$$

$$\triangle w_{ij} = -\alpha_2 \delta_{1j}(\boldsymbol{z}^l) z_i^l \tag{9}$$

$$(i = 0, \cdots, N, j = 1, \cdots, M)$$

where α_1 and α_2 are learning coefficients,

$$\delta_{1j}(\boldsymbol{z}) = (h(\boldsymbol{z}) - d(\boldsymbol{z})) h(\boldsymbol{z})(1 - h(\boldsymbol{z})) \tag{10}$$

and

$$\delta_{2j}(\boldsymbol{z}) = \delta_{1j} v_j e_j(\boldsymbol{z})(1 - e_j(\boldsymbol{z})). \tag{11}$$

Then, BP learning method is shown in Fig.5 [14].

III. BACK PROPAGATION LEARNING FOR SECURE MULTIPARTY METHOD

A. Perceptron learning for secure shared data

Let us consider a system composed of client and m servers(See Fig.1). In learning on cloud system, learning data and weight parameters are shared to each server in multiplication form. Each server updates shared weight parameters and sends the computation result to the client. The client can get new weight parameters by multiplying the results of m servers. This is verified from Eqs.(15) and (16). The process is iterated until the error (difference) between the output of the neuron and the desired output becomes sufficiently small. The problem is how the weight parameter w on the user is updated using the set of learning data shared on each server. The shared representation of learning data $\{(\boldsymbol{z}^l, d(\boldsymbol{z}^l)) | l \in Z_L\}$and parameters are given as follows:

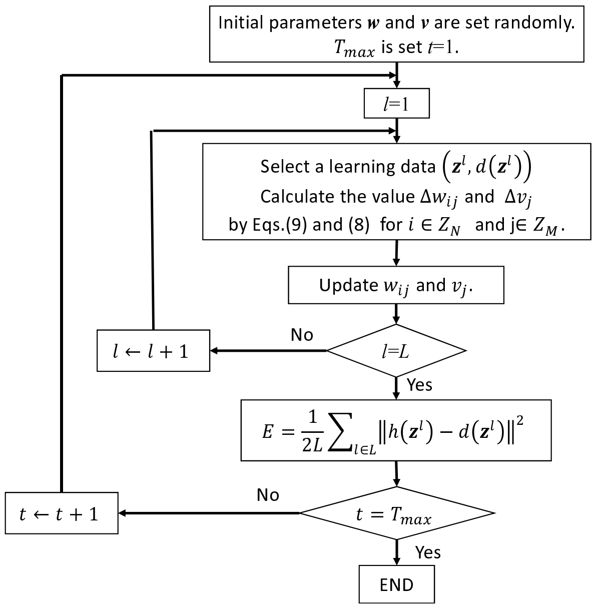
Initial parameters w and v are set randomly. T_{max} is set t=1.
l=1
Select a learning data $(z^l, d(z^l))$ Calculate the value Δw_{ij} and Δv_j by Eqs.(9) and (8) for $i \in Z_N$ and $j \in Z_M$.
Update w_{ij} and v_j.
l=L
No
Yes
$l \leftarrow l+1$
$E = \frac{1}{2L}\sum_{l\in L}\|h(z^l) - d(z^l)\|^2$
$t = T_{max}$
$t \leftarrow t+1$
END

Fig. 5. BP learning for three-layered neural network.

$$z^l = (z_1^l, \cdots, z_i^l, \cdots, z_N^l) \tag{12}$$

for $l \in Z_L$ and

$$z_i^l = \Pi_{k=1}^m (z_i^l)^k \tag{13}$$

for $i \in Z_N$,

$$d(z^l) = \sum_{k=1}^m (d(z^l))^k \tag{14}$$

Note that Eqs.(13) and (14) are in multiplication form for shared data.

In this case, Eqs.(4) and (5) are renewed as follows:

$$\triangle v_j^k = \frac{\partial E}{\partial v_j^k} = (g(z) - d(z)) z_j v_j / v_j^k \tag{15}$$

$$(v_j^k)(t+1) = (v_j^k)(t) + K \triangle v_j^k(t) \tag{16}$$

for $v = (v_1, \cdots, v_j, \cdots, v_M)$ and $v_j = \Pi_{k=1}^m v_j^k$.

Eq.(16) means that each server can update the weight by dividing by v_j^k for the conventional method.

The learning process of the perceptron learning is shown in TableII.

B. BP learning for secure shared data

In this section, we propose a BP learning method for secure multi-party computation as the same method as the section 3.1. The problem is how to the weights $\{w_{ij}\}$ and $\{v_j\}$ are updated using the set of learning data shared on each server. The shared representation of learning data and parameters are given as follows:

$$z^l = (z_1^l, \cdots, z_i^l, \cdots, z_N^l) \tag{17}$$

for $l \in Z_L^0$,

$$z_i^l = \Pi_{k=1}^m (z_i^l)^k \tag{18}$$

TABLE IV
THE INITIAL CONDITIONS FOR SIMULATIONS OF FUNCTION APPROXIMATION

	conventional method	proposed method
K_w	0.01	0.01
K_v	0.01	0.01
T_{max}	50000	50000
Initial w_{ij}	random on [0,1]	
Initial v_i	random on [0,1]	

for $i \in Z_N$,

$$\begin{aligned} d(z^l) &= \sum_{k=1}^m (d(z^l))^k & (19)\\ v &= (v_1, \cdots, v_j, \cdots, v_M) & (20)\\ v_j &= \Pi_{k=1}^m v_j^k & (21)\\ w &= (w_1, \cdots, w_j, \cdots, w_M) & (22)\\ w_j &= (w_{1j}, \cdots, w_{ij}, \cdots, w_{Nj}) & (23)\\ w_{ij} &= \Pi_{k=1}^m (w_{ij})^k & (24) \end{aligned}$$

Further, Eqs. (8) and (9) are renewed as follows:

$$\triangle v_j^k = -\alpha_1 \delta_{1j}(z^l) v_j e_j(z^l)/v_j^k \tag{25}$$

$$\triangle w_{ij}^k = -\alpha_2 \delta_{2j}(z^l) w_{ij} x_i^l / w_{ij}^k \tag{26}$$

Eqs.(25) and (26) mean that each server can update the weight by dividing by v_j^k and w_{ij}^k in the proposed method.

The learning process of the three-layered neural network is shown in TableIII.

C. Numerical Simulation for Function Approximation

In this section, numerical simulation of function approximation for the conventional and proposed methods is performed. The conventional method means the conventional BP method without data division and the proposed method is one with $m = 3$ and $m = 10$. In this case, the multiplication form is used as $a = A^{(1)} \cdots A^{(m)}$, where $A^{(k)}$ for $1 \le k \le m-1$ is a random number in $[0,1]$ or $[-1,1]$, and $A^{(m)} = a/(A^{(1)} \cdots A^{(m-1)})$. This simulation uses three systems specified by the following functions with 4-dimensional input space $[0,1]^4$ (for Eq.(27)) and $[-1,1]^4$ (for Eqs.(28) and (29)). The simulation condition is shown in Table IV. The numbers of learning and test data randomly selected are 512 and 6400, respectively.

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{12} \tag{27}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{446.52} \tag{28}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(3e^{3x_3} + 2e^{-4x_4})^{-0.5} - 0.077}{4.68} \tag{29}$$

Table V shows the results of comparison between the conventional and the proposed methods. In each box of Table V, three numbers from the top to the bottom show MSE of training ($\times 10^{-4}$), MSE of test ($\times 10^{-4}$) and the number

TABLE II
LEARNING PROCESS OF THE PERCEPTRON LEARNING FOR SMC.

	Client	k-th Server
Initial condition	The weight $\boldsymbol{v} = (v_1, \cdots, v_M)$ is selected randomly, and send $v_j^k (v_j = \Pi_{k=1}^m v_j^k)$ to each server for $k \in Z_m$ and $j \in Z_M$.	$\{(\boldsymbol{z}^l)^k \mid l \in Z_L\}$
Step 1	A number l is selected randomly	$\{v_j^k\}$ for $j \in Z_M$
Step 2		Compute $\mu_j^k = v_j^k (z_j^l)^k$ for $j \in Z_M$ and send μ_j^k and $(d(\boldsymbol{z}^l))^k$ to Client.
Step 3	Compute $\mu_j = \Pi_{k=1}^m \mu_j^k$ for $j \in Z_M$, $d(\boldsymbol{z}^l) = \sum_{k=1}^m \left(d(\boldsymbol{z}^l)\right)^k$ and $z_j v_j = \Pi_{k=1}^m (z_j^l)^k v_j^k$	
Step 4	Compute $\phi = K \left(g(\sum_{i=1}^N \mu_i) - d(\boldsymbol{z}^l)\right) z_j v_j$ and send ϕ to each server	
Step 5		Compute $\triangle v_j^k = \phi / v_j^k$ and $v_j^k \leftarrow v_j^k + \triangle v_j^k$ for $j \in Z_M$ and send them to Client
Step 6	If $t \neq T_{max}$ then go to Step 2 with $t \leftarrow t + 1$ else the algorithm terminates.	

TABLE III
LEARNING PROCESS OF BP FOR SMC.

	Client	k-th Server
Initial condition	The weight $\boldsymbol{v} = (v_1, \cdots, v_M)$ and $\boldsymbol{w}_j = (w_{1j}, \cdots, w_{ij}, \cdots, w_{Nj})$ for $j \in Z_M$ are selected randomly, $v_j = \Pi_{k=1}^m v_j^k$ and $w_{ij} = \Pi_{k=1}^m w_{ij}^k$. Send v_j^k and w_{ij}^k to each server.	$\{(\boldsymbol{z}^l)^k\}$ for $l \in Z_L$
Step 1	A number l is selected randomly and send it to Server.	$\{w_{ij}^k, v_j^k\}$ for $\in Z_N$ and $j \in Z_M$
Step 2		Compute $\mu_{ij}^k = w_{ij}^k (z_i^l)^k$ for $i \in Z_N$ and $j \in Z_M$. Send them to Client
Step 3	Compute $w_{ij} z_i = \Pi_{k=1}^m \mu_{ij}^k$, $h_j = \sum_{i=0}^L w_{ij} z_i$ and $y_j = e(h_j)$	
Step 4	Send y_j^k for $k \in Z_m$ to each server, where $y_j = \Pi_{k=1}^m y_j^k$	
Step 5		Compute $\phi_j^k = v_j^k y_j^k$ for $j \in Z_L$ and send it to Client
Step 6	Compute $v_j y_j = \Pi_{k=1}^m \phi_j^k$, $d(\boldsymbol{z}^l) = \sum_{k=1}^m (d(\boldsymbol{z}^l))^k$ $z_0^l = \sum_{j=0}^M v_j y_j$ and $z^l = g(z_0^l)$	
Step 7	Compute $\phi_1 = \left(g(\boldsymbol{z}^l) - d(\boldsymbol{z}^l)\right) g(\boldsymbol{z}^l)(1 - g(\boldsymbol{z}^l)) v_j y_j$ and $\phi_2 = \left(g(\boldsymbol{z}^l) - d(\boldsymbol{z}^l)\right) g(\boldsymbol{z}^l)(1 - g(\boldsymbol{z}^l)) v_j y_j (1 - y_j) w_{ij} z_i^l$ and send them to each server	
Step 8		Compute$v_j^k \leftarrow v_j^k + K_1 \phi_1 / v_j^k$ and $w_{ij}^k \leftarrow w_{ij}^k + K_2 \phi_2 / w_{ij}^k$ and send them to Client
Step 9	If $t \neq T_{max}$ then go to Step 1 with $t \leftarrow t + 1$ else the algorithm terminates.	

TABLE V
RESULT OF SIMULATION OF FUNCTION APPROXIMATION

	Eq.(27)	Eq.(28)	Eq.(29)
conventional method	0.80 1.00 121	2.06 2.66 121	1.22 1.41 121
proposed method ($m = 3$)	0.21 0.38 333	0.52 1.02 333	0.27 0.41 333
proposed method ($m = 10$)	0.35 0.54 1210	0.44 0.63 1210	0.99 1.63 1210

of parameters, respectively. The result of simulation is the average value from twenty trials.

The result shows that the accuracy between the conventional and the proposed methods is almost same. Further, it is shown that the results for $m = 3$ and $m = 10$ is also about the same accuracy.

IV. CONCLUSION

The secure multi-party computation (SMC) is one of secure sharing and computing methods. The SMC with secure shared data in addition and multiplication forms is known, but complex computation processing such as BP learning has never proposed yet. In this paper, we proposed a learning method for SMC on cloud computing system and proved the validity of it. Further, the performance of the proposed method was shown in numerical simulations.

In the future work, we consider the improved method to reduce the computation of client and develop AUI (Application User Interface) for the client.

REFERENCES

[1] C. C. Aggarwal, and P. S. Yu, Privacy-Preserving Data Mining: Models and Algorithms, ISBN 978-0-387-70991-8, Springer-Verlag, 2009.
[2] S. Subashini, and V. Kavitha, A survey on security issues in service delivery models of cloud computing, J. Network and Computer Applications, Vol.34,pp.1-11, 2011.
[3] C. Gentry, Fully Homomorphic Encryption Using Ideal Lattices, STOC2009, pp.169-178, 2009.
[4] HElib, An Implementation of homomorphic encryption, https://github.com/shaih/HElib
[5] A. Shamir, How to share a secret, Comm. ACM, Vol. 22, No. 11, pp. 612-613, 1979.
[6] A. Beimel, Secret-sharing schemes: a survey, in Proc. of the Third international conference on Coding and cryptology (IWCC 11), 2011.
[7] R. Canetti, et al., Adaptively secure multi-party computation, STOC' 96, pp. 639-648, 1996.
[8] R. Cramer, et al., General secure multi-party computation from any linear secret-sharing scheme, EUROCRYPT' 00, 2000.
[9] A. Ben-David, et al., Fair play MP: a system for secure multi-party computation, ACM CCS' 08, 2008.
[10] K. Chida, et al., A Lightweight Three-Party Secure Function Evaluation with Error Detection and Its Experimental Result, IPSJ Journal Vol. 52 No. 9, pp. 2674-2685, Sep. 2011(in Japanese).
[11] O. Catrina, et al., Secure multiparty linear programming using fixed point arithmetic, ESORICS 2010, 2010.
[12] Y. Miyanishi, A. Kanaoka, F. Sato, X. Han, S. Kitagami, Y.Urano, N. Shiratori, New Methods to Ensure Security to Increase User's Sense of Safety in Cloud Services, Proc. of The 14th IEEE Int. Conference on Scalable Computing and Communications (ScalCom-2014), pp.859-865, Bali, Dec.2014.
[13] H. Miyajima, N. Shigei, H. Miyajima, Performance Comparison of Hybrid Electromagnetism-like Mechanism Algorithms with Descent Method, JAISCR,vol.5, no.4, 2015.
[14] M. M. Gupta, L. Jin and N. Homma, Static and Dynamic Neural Networks, IEEE Press, 2003.