Advances in Fuzzy Sets and Systems

Published Online: November 2015
http://dx.doi.org/10.17654/AFSSDec2015_155_175
Volume 20, Number 2, 2015, Pages 155-175 **ISSN: 0973-421X**

SOME PROPERTIES ON FUZZY INFERENCE SYSTEMS COMPOSED OF SMALL NUMBER OF INPUT RULE MODULES

Hirofumi Miyajima, Noritaka Shigei and Hiromi Miyajima

Graduate School of Science and Engineering
Kagoshima University
1-21-40, Korimoto, Kagoshima, Japan
e-mail: k3768085@kadai.jp
shigei@eee.kagoshima-u.ac.jp
miya@eee.kagoshima-u.ac.jp

Abstract

The automatic construction of fuzzy system with a large number of input variables involves many difficulties such as large time complexity and getting stuck in a shallow and local minimum. As models to overcome them, the SIRMs (single input rule modules) and DIRMs (double input rule modules) models have been proposed. In some numerical simulations such as EX-OR and control problems, it has been shown that there exists the difference of the capability between DIRMs and SIRMs models. In this paper, we theoretically show the difference of the capability among SNIRMs (small number of input rule modules) models. As a result, there exists the difference of the capability among SNIRMs models. Further, in order to reduce the number of modules of the SNIRMs model effectively, we propose two types of learning algorithms. The one, called pruning algorithm

Received: March 6, 2015; Accepted: April 20, 2015
2010 Mathematics Subject Classification: 68-XX.
Keywords and phrases: fuzzy inference system, SIRMs model, SNIRMs model, generative algorithm, pruning algorithm.

for learning, deletes some rule modules in the SNIRMs model. The other one, called generative algorithm for learning, incrementally adds some modules to the SNIRMs model. The simulation result shows that the SNIRMs models obtained by the proposed algorithms are superior in terms of accuracy compared to the conventional SIRMs model and in terms of the number of modules to the conventional SNIRMs model.

I. Introduction

Many studies on self-tuning fuzzy systems [1-5] have been made. The aim of these studies is to construct automatically fuzzy inference rules from input and output data. The steepest descend method is usually used [6-8]. Obvious drawbacks of the steepest descend method are its large computational complexity and getting stuck in a shallow and local minimum. Further, there is a problem that it is difficult to apply learning for fuzzy inference systems to the problem with a large number of inputs. In order to overcome them, some novel methods have been developed which (1) create fuzzy rules one by one starting from any number of rules [9], (2) delete fuzzy rules one by one starting from a sufficiently large number of rules [10], (3) use GA and PSO to determine the structure of the fuzzy model [7, 11], (4) use a self-organization or a vector quantization technique to determine the initial assignment of fuzzy rules [12], and (5) use generalized objective functions [13]. However, there are little effective models of fuzzy inference systems dealing with a large number of input variables; in most of the conventional methods, fuzzy inference systems deal with a small number of input variables. Further, there is a problem of trade-off between interpretability and accuracy [4, 15]. SIRMs model aims to obtain a better solution by using fuzzy inference systems composed of single input rule modules, where the output is determined as the weighted sum of all modules [16, 17]. However, it is known that SIRMs model does not always achieve good performance in nonlinear system. Therefore, we have proposed SNIRMs model as a generalized SIRMs model, in which each module is composed of small number of input variables [18-20]. In particular, DIRMs

model has been proposed as effective one of SNIRMs models. Then, there is a question how is the difference of the capability among SNIRMs model. We have already shown that the EX-OR with two variables and control problems can be implemented by DIRMs model but not by SIRMs model in numerical simulation [20]. However, we do not study about the difference of the capability among SNIRMs models. Further, though the SNIRMs model has simple structure, it needs more modules than the SIRMs model.

In this paper, we show the difference of the capability among SNIRMs models theoretically. Further, in order to reduce the number of modules of the SNIRMs model effectively, we propose two types of learning algorithms. The one, called *pruning algorithm* for learning, delete some rule modules in the SNIRMs model. The other one, called *generative algorithm* for learning, incrementally adds some modules to the SNIRMs model. The simulation result shows that the SNIRMs models obtained by the proposed learning algorithms are superior in terms of accuracy compared to the conventional SIRMs model and in terms of the number of modules to the conventional SNIRMs model.

II. Fuzzy Inference Model and its Learning

The conventional fuzzy inference model using the steepest descend method is described [1-3]. Let $Z_j = \{1, ..., j\}$ and $Z_j^* = \{0, 1, ..., j\}$ for the positive integer j. Let $\boldsymbol{R}$ be the set of real numbers. Let $\boldsymbol{x} = (x_1, ..., x_m)$ and y^r be input and output data, respectively, where $x_i \in \boldsymbol{R}$ for $i \in Z_m$ and $y^r \in \boldsymbol{R}$. Then the rule of simplified fuzzy inference model is expressed as

$$R_j : \text{if } x_1 \text{ is } M_{1j} \text{ and} \cdots \text{and } x_m \text{ is } M_{mj}, \text{ then } y \text{ is } w_j, \qquad (1)$$

where $j \in Z_n$ is a rule number, $i \in Z_m$ is a variable number, M_{ij} is a membership function of the antecedent part, and w_j is the weight of the consequent part.

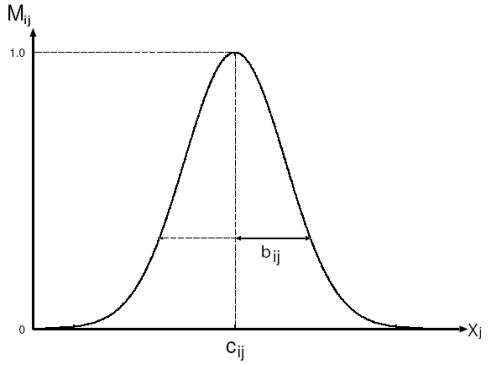

Figure 1. The Gaussian membership function.

A membership value of the antecedent part μ_j for input x is expressed as

$$\mu_j = \prod_{i=1}^{m} M_{ij}(x_i). \tag{2}$$

Let c_{ij} and b_{ij} denote the center and the width values of M_{ij}, respectively. If Gaussian membership function is used, then M_{ij} is expressed as follows (see Figure 1):

$$M_{ij} = \exp\left(-\frac{1}{2}\left(\frac{x_j - c_{ij}}{b_{ij}}\right)^2\right). \tag{3}$$

The output y^* of fuzzy inference is calculated by the following equation:

$$y^* = \frac{\sum_{j=1}^{n} \mu_j \cdot w_j}{\sum_{j=1}^{n} \mu_j}. \tag{4}$$

The objective function E is defined to evaluate the inference error between the desirable output y^r and the inference output y^*.

$$E = \frac{1}{2}(y^* - y^r)^2. \tag{5}$$

In order to minimize the objective function E, the parameters $\alpha \in \{c_{ij}, b_{ij}, w_{ij}\}$ are updated based on the descent method as follows [1-3]:

$$\alpha(t+1) = \alpha(t) - K_\alpha \frac{\partial E}{\partial \alpha}, \tag{6}$$

where t is iteration time and K_α is a constant. When the Gaussian membership function as shown in Figure 1 is used as the membership function, the following relation holds:

$$\frac{\partial E}{\partial c_{ij}} = \frac{\mu_j}{\sum_{j=1}^{n} \mu_j} \cdot (y^* - y^r) \cdot (w_j - y^*) \cdot \frac{x_j - c_{ij}}{b_{ij}^2}, \tag{7}$$

$$\frac{\partial E}{\partial b_{ij}} = \frac{\mu_j}{\sum_{j=1}^{n} \mu_j} \cdot (y^* - y^r) \cdot (w_j - y^*) \cdot \frac{(x_j - c_{ij})^2}{b_{ij}^3}, \tag{8}$$

$$\frac{\partial E}{\partial w_j} = \frac{\mu_j}{\sum_{j=1}^{n} \mu_j} \cdot (y^* - y^r). \tag{9}$$

In this section, we describe the conventional learning algorithm.

Let $\boldsymbol{D} = \{(x_1^p, ..., x_m^p, y_p^r) \mid p \in Z_P\}$ be the set of learning data. The objective of learning is minimizing the following mean square error (MSE):

$$E = \frac{1}{P} \sum_{p=1}^{P} (y_p^* - y_p^r)^2. \tag{10}$$

The conventional learning algorithm is shown below [1-3, 9].

Learning Algorithm A

Step A1 [Initialization]**.** The threshold θ of inference error and the maximum number of learning time $T_{\max}$ are given. The number of rules n is set to n_0. Let $t = 1$.

Step A2. The parameters b_{ij}, c_{ij} and w_i are set to the initial values.

Step A3. Let $p = 1$.

Step A4. A data $(x_1^p, ..., x_m^p, y_p^r) \in \boldsymbol{D}$ is given.

Step A5. From equations (2) and (4), μ_i and y^* are computed.

Step A6. Parameters c_{ij}, b_{ij} and w_i are updated by equations (6), (7), (8) and (9).

Step A7. If $p = P$, then go to Step A8 and if $p < P$, then go to Step A4 with $p \leftarrow p + 1$.

Step A8. Let $E(t)$ be inference error at step t calculated by equation (10). If $E(t) > \theta$ and $t < T_{\max}$, then go to Step A3 with $t \leftarrow t + 1$ else if $E(t) \leq \theta$ and $t \leq T_{\max}$, then the algorithm terminates.

Step A9. If $t > T_{\max}$ and $E(t) > \theta$, then go to Step A2 with $n \leftarrow n + 1$ and $t = 1$.

III. The SNIRMs and DIRMs Models

The SNIRMs, SIRMs, DIRMs, and TIRMs (triple input rule modules) models are introduced [19]. Let U_k^m be the set of all ordered *k*-tuples of Z_m, that is

$$U_k^m = \{l_1 \cdots l_k \,|\, l_i < l_j \text{ if } i < j\}. \tag{11}$$

Example 1. $U_2^3 = \{12, 13, 23\}$, $U_1^3 = \{1, 2, 3\}$, $U_3^4 = \{123, 124, 234, 134\}$, $U_2^4 = \{12, 13, 14, 23, 24, 34\}$, $U_1^4 = \{1, 2, 3, 4\}$.

Then each rule of SNIRMs model for U_k^m is defined as follows:

SNIRM-$l_1 \cdots l_k$:

$$\{R_i^{l_1 \cdots l_k} : \text{if } x_{l_1} \text{ is } M_i^{l_1} \text{ and} \cdots \text{and } x_{l_k} \text{ is } M_i^{l_k}, \text{ then } y_{l_1 \cdots l_k} \text{ is } w_i^{l_1 \cdots l_k}\}_{i=1}^n. \tag{12}$$

The SNIRMs model composed of rule modules with k variables is called *k-SNIRMs model*. The cases of 1-SNIRMs, 2-SNIRMs, and 3-SNIRMs are also called *SIRMs*, *DIRMs* and *TIRMs* ones, respectively.

Example 2. For U_1^3 and U_2^3, the SIRMs and DIRMs models are obtained as follows:

SIRM–1. $\{R_i^1$: if x_1 is M_i^1, then y_1 is $w_i^1\}_{i=1}^n$,

SIRM–2. $\{R_i^2$: if x_2 is M_i^2, then y_2 is $w_i^2\}_{i=1}^n$,

SIRM–3. $\{R_i^3$: if x_3 is M_i^3, then y_3 is $w_i^3\}_{i=1}^n$,

DIRM–12. $\{R_i^{12}$: if x_1 is M_i^1 and x_2 is M_i^2, then y_{12} is $w_i^{12}\}_{i=1}^n$,

DIRM–13. $\{R_i^{13}$: if x_1 is M_i^1 and x_3 is M_i^3, then y_{13} is $w_i^{13}\}_{i=1}^n$,

DIRM–23. $\{R_i^{23}$: if x_2 is M_i^2 and x_3 is M_i^3, then y_{23} is $w_i^{23}\}_{i=1}^n$.

Figure 2 shows the relation among the conventional, SIRMs and DIRMs models for $m = 3$. Example 2 shows DIRMs and SIRMs models for $m = 3$.

Let $x = (x_1, ..., x_m)$ and y^r be input and output data, respectively. The fitness of the ith rule and the output of SNIRM-$l_1 \cdots l_k$ are as follows:

$$\mu_i^{l_1 \cdots l_k} = M_i^{l_1}(x_{l_1}) M_i^{l_2}(x_{l_2}) \cdots M_i^{l_k}(x_{l_k}), \tag{13}$$

$$y_{l_1 \cdots l_k}^o = \frac{\sum_{i=1}^n \mu_i^{l_1 \cdots l_k} w_i^{l_1 \cdots l_k}}{\sum_{i=1}^n \mu_i^{l_1 \cdots l_k}}. \tag{14}$$

In this model, in addition to the conventional parameters c, b and w, the importance degree h is introduced. Let h_L be the importance degree of each module L, where $L = l_1 \cdots l_k$.

$$y^* = \sum_{L \in U_k^m} h_L \cdot y_L^o. \tag{15}$$

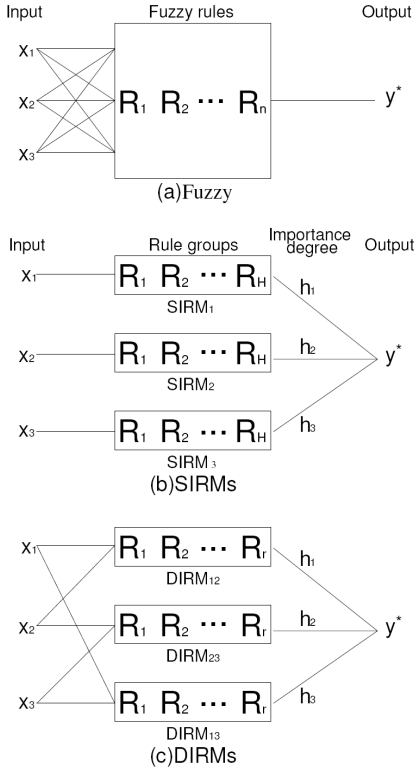

Figure 2. The relation among the conventional, SIRMs and DIRMs models for $m = 3$.

As the same method as the equations (2) to (5), $\frac{\partial E}{\partial \alpha}$'s are calculated as follows:

$$\frac{\partial E}{\partial h_L} = (y^* - y^r) y_L^o, \tag{16}$$

$$\frac{\partial E}{\partial w_i^L} = h_L \cdot \frac{\mu_i^L}{\sum_{i=1}^n \mu_i^L} (y^* - y^r), \tag{17}$$

$$\frac{\partial E}{\partial c_{rl_j}^L} = (y^* - y^r) h_L \frac{w_r^L - y_L^o}{\sum_{r=1}^n \mu_r^L} \frac{x_{rl_j} - c_{rl_j}^L}{(b_{rl_j}^L)^2}, \tag{18}$$

$$\frac{\partial E}{\partial b_{rl_j}^L} = (y^* - y^r) h_L \frac{w_r^L - y_L^o}{\sum_{r=1}^n \mu_r^L} \frac{(x_{rl_j} - c_{rl_j}^L)^2}{(b_{rl_j}^L)^3}. \tag{19}$$

It is known that the SIRMs model does not always achieve good performance in nonlinear problems [18-20]. On the other hand, when the number of input variables is large, Algorithm A requires a large time complexity and tends to easily get stuck into a shallow local minimum. The DIRMs and TIRMs models can achieve good performance in nonlinear problems compared to the SIRMs model and is simpler than the conventional model.

Let ***D*** be the set of learning data. A learning algorithm for *k*-SNIRMs model is given as follows:

Learning Algorithm B(k)

1 [Initialization]**.** The initial parameters, c_i^L, b_i^L, w_i^L, h^L, θ_1, $T_{\max}$, K_c, K_b and K_w for $L \in U_k^m$ are set. Let $t = 1$.

2. Let $p = 1$.

3 [Learning data]**.** An input and output data $(x_1^p, ..., x_m^p, y_p^r) \in \boldsymbol{D}$ is given.

4 [Output of fuzzy inference]**.** Membership value of each rule is calculated by equation (13). Inference output y_p^* is calculated by equations (14) and (15).

5 [Updating parameters]**.** Importance degree h_L is updated by equation (16). Real number w_i^L is updated by equation (17). Parameters c_i^L and b_i^L are updated by equations (18) and (19), respectively.

6 [Termination]**.** If $p = P$, then go to the next step. If $p < P$, then $p \leftarrow p + 1$ and go to Step 3.

7. Inference error $E(t)$ is calculated by equation (10). If $E(t) < \theta_1$, then learning is terminated.

8. If $t \neq T_{\max}$, then $t \leftarrow t+1$ and go to Step 2. Otherwise learning is terminated.

Note that the numbers of rules for the conventional, DIRMs and SIRMs models are $O(H^m)$, $O(m^2H^2)$ and $O(mH)$, respectively, where H is the number of fuzzy partitions.

In order to reduce the number of rule for DIRMs model, we propose generative algorithm for learning (GAL) that incrementally adds m modules to the SIRMs model. The model is composed of SIRMs and DIRMs models. Let q be the target number of modules. The algorithm is as follows:

Learning Algorithm C(q) (GAL for DIRMs model)

Step 1. Algorithm B for $k = 1$ is performed. SIRMs model is performed. SIRMs model is constructed. Let $i = 1$ and $j = 0$.

Step 2. Select a variable x_0 with the i-th high importance degree in Step 1 and add all new modules composed of two input variables including the variable x_0 to the system, where $1 \leq i \leq m$.

Step 3. In order to adjust the parameters of the system, Algorithm B is performed. If $j = q$, then the algorithm terminates. Otherwise go to Step 2 as $i = i+1$ and $j = j + (m - i)$.

Further, we propose pruning algorithm for learning (PAL) that eliminates modules until reaching the target number of modules q. The algorithm is as follows:

Learning Algorithm D(q) (PAL for DIRMs model)

Step 1. Let i the number of modules of the constructive model and let $i =_m C_2$.

Step 2. Learning of the DIRMs model with *i* modules is performed using Algorithm B.

Step 3. If there exists an input variable excluding in all modules for the constructed model, the algorithm terminates.

Step 4. If $i = q$, then the algorithm terminates and the model with q modules is obtained.

Step 5. Delete one module with the lowest importance degree from all rule modules of the DIRMS model. Let $i = i - 1$ and go to Step 2.

The other methods are also considered with pruning algorithm such as the method using forgetting one [19]. In our simulations, there exists no difference between them.

Here, we proposed the GAL and PAL for DIRMs model. The proposed methods are also applied to the cases of TIRMs and other SNIRMs models easily.

IV. The Capability of SNIRMs Models and Numerical Simulations

A. The capability of SNIRMs models

Let us consider the capability of SNIRMs models using EX-OR problem with *m* variables. The EX-OR problem is defined as follows:

$$y = x_1 \oplus \cdots \oplus x_m,$$

where x_1, ..., x_m, $y \in \{0, 1\}$, and $\oplus$ means the exclusive OR operation [3].

Then, the following result holds:

Proposition. *The EX-OR problem with* $k+1$ *variables cannot be implemented by k-SNIRMs model.*

Proof. Let us show the case of $k = 2$ without loss of generality. Assume that there exists DIRMs (2-SNIRMs) model implementing the EX-OR problem with 3 variables as shown in Table I.

Table I. EX-OR function with $m = 3$

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

From the relation between the output of the model and Table I, the following relation holds:

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^2(0) w_i^{12}}{\sum_{i=1}^{n} M_i^1(0) M_i^2(0)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^3(0) w_i^{13}}{\sum_{i=1}^{n} M_i^1(0) M_i^3(0)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(0) M_i^3(0) w_i^{23}}{\sum_{i=1}^{n} M_i^2(0) M_i^3(0)} < 0.5, \tag{20}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^2(0) w_i^{12}}{\sum_{i=1}^{n} M_i^1(0) M_i^2(0)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^3(1) w_i^{13}}{\sum_{i=1}^{n} M_i^1(0) M_i^3(1)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(0) M_i^3(1) w_i^{23}}{\sum_{i=1}^{n} M_i^2(0) M_i^3(1)} \geq 0.5, \tag{21}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^2(1) w_i^{12}}{\sum_{i=1}^{n} M_i^1(0) M_i^2(1)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^3(0) w_i^{13}}{\sum_{i=1}^{n} M_i^1(0) M_i^3(0)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(1) M_i^3(0) w_i^{23}}{\sum_{i=1}^{n} M_i^2(1) M_i^3(0)} \geq 0.5, \tag{22}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^2(1) w_i^{12}}{\sum_{i=1}^{n} M_i^1(0) M_i^2(1)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(0) M_i^3(1) w_i^{13}}{\sum_{i=1}^{n} M_i^1(0) M_i^3(1)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(1) M_i^3(1) w_i^{23}}{\sum_{i=1}^{n} M_i^2(1) M_i^3(1)} < 0.5, \tag{23}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^2(0) w_i^{12}}{\sum_{i=1}^{n} M_i^1(1) M_i^2(0)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^3(0) w_i^{13}}{\sum_{i=1}^{n} M_i^1(1) M_i^3(0)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(0) M_i^3(0) w_i^{23}}{\sum_{i=1}^{n} M_i^2(0) M_i^3(0)} \geq 0.5, \tag{24}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^2(0) w_i^{12}}{\sum_{i=1}^{n} M_i^1(1) M_i^2(0)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^3(1) w_i^{13}}{\sum_{i=1}^{n} M_i^1(1) M_i^3(1)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(0) M_i^3(1) w_i^{23}}{\sum_{i=1}^{n} M_i^2(0) M_i^3(1)} < 0.5, \tag{25}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^2(1) w_i^{12}}{\sum_{i=1}^{n} M_i^1(1) M_i^2(1)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^3(0) w_i^{13}}{\sum_{i=1}^{n} M_i^1(1) M_i^3(0)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(1) M_i^3(0) w_i^{23}}{\sum_{i=1}^{n} M_i^2(1) M_i^3(0)} < 0.5, \tag{26}$$

$$h_1 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^2(1) w_i^{12}}{\sum_{i=1}^{n} M_i^1(1) M_i^2(1)} + h_2 \frac{\sum_{i=1}^{n} M_i^1(1) M_i^3(1) w_i^{13}}{\sum_{i=1}^{n} M_i^1(1) M_i^3(1)} + h_3 \frac{\sum_{i=1}^{n} M_i^2(1) M_i^3(1) w_i^{23}}{\sum_{i=1}^{n} M_i^2(1) M_i^3(1)} \geq 0.5. \tag{27}$$

Let each sum of equations (20), (24), (25) and (26), and equations (21), (22), (23) and (27) be denoted by A and B, respectively. Then, though the left hand sides of A and B are equal to each other, the right hand sides of A and B are not equal to each other. That is contradiction. As a result, there does not exist any DIRMs model that can implement EX-OR problem.

Corollary [20]**.** *The EX-OR problem with* 2 *variables cannot be implemented by any SIRMs model*.

Generally speaking, $(k+1)$-SNIRMs model has higher ability than *k*-SNIRMs model. On the other hand, SIRMs model can perform AND and OR logical functions. We conjecture that SIRMs model has the property of linearly separable as in the case of perceptron model [3]. The result is not proved yet.

In the following, let us show numerical simulations of logical functions as follows:

$$y = x_1 \vee \cdots \vee x_m,$$

$$y = x_1 \wedge \cdots \wedge x_m,$$

where $x_1, ..., x_m, y \in \{0, 1\}$ and $\vee$ and $\wedge$ mean OR and AND operators, respectively [3]. Let us implement OR, AND, and EX-OR operators. Table II shows the conditions for simulation. Table III shows the result of MSE for each model. It is shown that the EX-OR with $m = 3$ cannot be implemented by DIRMs model and the EX-OR with $m = 2$ cannot be implemented by SIRMs model.

Table II. The initial conditions for simulations of logical functions

	SIRMs		DIRMs		TIRMs
	OR, AND	EX-OR	OR, AND	EX-OR	EX-OR
K_c	0.0001	0.001	0.0001	0.001	0.001
K_b	0.0001	0.001	0.0001	0.001	0.001
K_w	0.1	0.01	0.1	0.01	0.01
K_h	0.5	0.05	0.5	0.05	0.05
T_{max}	100	100	1000	2000	10000
Initial c_{ij}	Equal intervals				
Initial b_{ij}	$\frac{1}{2(H-1)} \times (the\ domain\ of\ input)$				
Initial w_{ij}	Random on [0,1]				
Initial h_j	Random on [0,1]				

Table III. The result for simulation of logical functions

	SIRMs	DIRMs	TIRMs
OR($m = 2$)	0.00	0.00	
AND($m = 2$)	0.00	0.00	
EX-OR($m = 2$)	0.25	0.00	
EX-OR($m = 3$)		0.17	0.00

Further, let us consider about the universal approximation capability of the *k*-SNIRMs models for continuous functions. Let us consider the following continuous function:

$$f(x_1, x_2) = x_1 + x_2 - 2x_1x_2,$$

where $x_1, x_2, f(x_1, x_2) \in [0, 1]$. Remark that $f(0, 0) = f(1, 1) = 0$ and $f(0, 1) = f(1, 0) = 1$.

When we simulate the function $f(x_1, x_2)$ by DIRMs and SIRMs models, the MSE for DIRMs model decreases as *n* increases, but the MSE for SIRMs model does not always continue to decrease. It seems that SIRMs model does not have universal approximation capability because there exits a continuous function that cannot be implemented with arbitrary accuracy by SIRMs model. To be precise, it must be proved theoretically. From the same consideration, it seems that *k*-SNIRMs models do not have universal approximation capability for continuous functions, where $1 \le k \le m - 1$. It

is clear that *k*-SNIRMs models do not satisfy Stone-Weierstrass theorem [3] though it is sufficient condition for universal approximation capability. It is well known that the conventional models have universal approximation capability for continuous functions [3].

B. Numerical simulations

The numerical simulations for function approximation and pattern classification are performed. In the following, A, B, C, and D mean the conventional model by Algorithm A, the *k*-SNIRMs models by Algorithm B, the proposed methods by Algorithms C (GAL) and D (PAL), respectively.

(1) *Function approximation*. From Proposition in 4.1, it is shown that there exists the difference of the capabilities among the SNIRMs models. The following simulation examines how is the difference in approximation of continuous functions. This simulation uses four systems specified by the following functions with 4-dimensional input space $[0, 1]\times[0, 1]\times[0, 1]\times[0, 1]$.

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{12}, \tag{28}$$

$$y = \frac{\sin(2\pi x_1)\times\cos(x_2)\times\sin(\pi x_3)\times x_4 + 1.0}{2.0}, \tag{29}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{4\sin(\pi x_3) + 2\cos(\pi x_4) + 6}{446.52}, \tag{30}$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(3e^{3x_3} + 2e^{-4x_4})^{-0.5} - 0.077}{4.68}. \tag{31}$$

The simulation condition is shown in Table IV. The numbers of learning and test data randomly selected are 512 and 6400, respectively. Table V shows the results on comparison among the conventional, TIRMs, DIRMs and SIRMs models, where "#" means "the number of". Table VI shows the results on comparison among DIRMs, its generative and pruning algorithms. The result of simulation is the average value from ten trials.

The simulation results of function approximation show the following:

(1) From Table V, the conventional and TIRMs models are superior in accuracy to SIRMs and DIRMs ones and need parameters so much.

(2) From Table V, DIRMs model is superior in accuracy to SIRMs model and does not need parameters so much compared with SIRMs one.

(3) From Table VI, the model for PAL is superior in accuracy to the model for PAL and the model for GAL is superior in accuracy to the SIRMs model, that is, GAL and PAL achieve the high accuracy without using many parameters.

Table IV. The initial conditions for simulations of function approximation

	A	B($k=1$)	B($k=2$)	C	D
K_c	0.001	0.001	0.001	0.001	0.001
K_b	0.001	0.001	0.001	0.001	0.001
K_w	0.01	0.01	0.01	0.01	0.01
K_h		0.05	0.05	0.05	0.05
T_{max}	50000	100	2000	10000	20000

Initial c_{ij}	Equal intervals
Initial b_{ij}	$\frac{1}{2(H-1)} \times (the\ domain\ of\ input)$
Initial w_{ij}	Random on [0,1]
Initial h_j	Random on [0,1]

Table V. The result of simulation for function approximation by the conventional, SIRMs, DIRMs and TIRMs methods

Learning Algorithm		A	B ($k=1$)	B ($k=2$)	B ($k=3$)
Eq.(28)	# Parameters	729	40	276	760
	MSE of training($\times 10^{-3}$)	0.0045	1.052	0.026	0.0036
	MSE of test($\times 10^{-3}$)	0.0189	1.199	0.050	0.0133
Eq.(29)	# Parameters	729	40	276	760
	MSE of training($\times 10^{-3}$)	0.011	9.258	2.229	0.108
	MSE of test($\times 10^{-3}$)	0.114	10.567	7.456	1.582
Eq.(30)	# Parameters	729	40	276	760
	MSE of training($\times 10^{-3}$)	0.027	6.027	0.234	0.019
	MSE of test($\times 10^{-3}$)	0.234	6.682	0.615	0.151
Eq.(31)	# Parameters	729	40	276	760
	MSE of training($\times 10^{-3}$)	0.028	5.875	0.014	0.015
	MSE of test($\times 10^{-3}$)	0.163	7.190	0.030	0.064

Table VI. The results of simulation for function approximation by GAL and PAL

Learning Algorithm		B ($k=2$)	C($q=3$)	D ($q=3$)	D ($q=4$)
Eq.(28)	# Parameters	276	178	138	184
	MSE of training($\times 10^{-3}$)	0.026	0.443	0.431	0.154
	MSE of test($\times 10^{-3}$)	0.050	0.686	0.835	0.275
Eq.(29)	# Parameters	276	178	138	184
	MSE of training($\times 10^{-3}$)	2.23	4.75	2.60	2.25
	MSE of test($\times 10^{-3}$)	7.46	7.41	6.48	7.30
Eq.(30)	# Parameters	276	178	138	184
	MSE of training($\times 10^{-3}$)	0.234	1.37	0.014	0.010
	MSE of test($\times 10^{-3}$)	0.615	1.98	0.020	0.016
Eq.(31)	# Parameters	276	178	138	184
	MSE of training($\times 10^{-3}$)	0.014	1.46	1.32	0.49
	MSE of test($\times 10^{-3}$)	0.030	0.96	2.30	0.83

Table VII. Data for pattern classification

	# input	# clusters	# data
Iris	4	3	150
Wine	13	3	178
BCW	9	2	683

Table VIII. The initial conditions for simulation of pattern classification

	B(k=1)	B($k=2$)	C	D
K_c	0.001	0.001	0.001	0.01
K_b	0.001	0.001	0.001	0.01
K_w	0.01	0.01	0.05	0.1
K_h	0.05	0.05	0.05	0.1
T_{max}	100	1000	500	20000
Initial c_{ij}	Equal intervals			
Initial b_{ij}	$\frac{1.5}{2(H-1)} \times (the\ domain\ of\ input)$			
Initial w_{ij}	Random on [0,1]			
Initial h_j	Random on [0,1]			

(2) *Pattern classification*. As the SNIRMs models such as SIRMs and DIRMs have few parameters, it is possible to implement the problems with a large number of input variables. Here, the proposed methods are applied to pattern classification shown in Table VII [21]. Wine and BCW data have comparatively high dimension of input, so it is difficult to implement by the conventional methods. The simulation condition is shown in Table VIII.

The rate of 2/3 for all data is used for learning and the rate of 1/3 is used for test. Table IX shows the result for simulations. In Table IX, error rate means the rate of the number of misclassified data for all data. The result of simulation is average value from ten trials.

The results of pattern classification show the following:

(1) It is shown that three benchmark problems are applied by all models excluding the case of Wine data for Algorithm B $(k = 1)$.

(2) The models for GAL and PAL do not need so much parameters to implement the problems.

(3) The proposed method can be implemented to the problems with comparatively high dimension of input.

(4) The conventional model can be applied to Iris data and the error rate is about 4.0%, that is, the result is almost the same as the case of the obtained result in Table IX.

Table IX. Simulation result of pattern classification

Learning Algorithm		B $(k = 1)$	B $(k = 2)$	C	D
Iris	# Parameters	40	276	178$(q = 3)$	138$(q = 3)$
	Error Rate of training$(\times 10^{-2})$	4.80	2.80	0.93	0.67
	Error Rate of test$(\times 10^{-2})$	5.17	1.199	3.54	4.67
Wine	# Parameters	130	3588	694$(q = 12)$	1196$(q = 26)$
	Error Rate of training$(\times 10^{-2})$	6.00	0.00	2.22	3.92
	Error Rate of test$(\times 10^{-2})$	27.8	12.2	12.8	9.04
BCW	# Parameters	90	1656	458$(q = 8)$	368$(q = 8)$
	Error Rate of training$(\times 10^{-2})$	1.28	0.67	2.16	3.22
	Error Rate of test$(\times 10^{-2})$	3.90	4.50	3.87	5.14

V. Conclusions

In the previous paper, DIRMs model as a generalized SIRMs model has been proposed and it is shown that DIRMs model is superior in accuracy to SIRMs model in numerical simulations. But, there are few results on the relation among SNIRs models. In this paper, it is clarified that there exists the difference of the capability among SNIRMs methods using logical functions. That is, there exists a problem that can be implemented by $(k+1)$-SNIRMs model but not by k-SNIRMs model theoretically. Further, in order to compare the approximating capability for continuous functions,

numerical simulations for four functions have been performed. As the results, it is shown that DIRMs and TIRMs models have high approximation capability, though they need fewer parameters than the conventional models and the models for GAL and PAL perform sufficient approximation capability compared with SIRMs models. Furthermore, it is shown that the problem with a large number of input variables can be implemented by DIRMs and SIRMs models, but it is difficult to implement by the conventional model. With the same problem, the models for GAL and PAL are superior in accuracy to SIRMs model. In the future work, we will consider the theoretical results on SNIRMs models.

References

[1] B. Kosko, Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence, Prentice Hall, Englewood Clis, NJ, 1992.

[2] C. Lin and C. Lee, Neural Fuzzy Systems, Prentice Hall, PTR, 1996.

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

[4] J. Casillas, O. Cordon, F. Herrera and L. Magdalena, Accuracy improvements in linguistic fuzzy modeling, Studies in Fuzziness and Soft Computing, Vol. 129, Springer, 2003.

[5] B. Liu, Theory and practice of uncertain programming, Studies in Fuzziness and Soft Computing, Vol. 239, Springer, 2009.

[6] H. Nomura, I. Hayashi and N. Wakami, A self-tuning method of fuzzy reasoning by delta rule and its application to a moving obstacle avoidance, Journal of Japan Society for Fuzzy Theory and Systems 4(2) (1992), 379-388 (in Japanese).

[7] H. Nomura, I. Hayashi and N. Wakami, A learning method of simplified fuzzy reasoning by genetic algorithm, Proc. of the Int. Fuzzy Systems and Intelligent Control Conference, 1992, pp. 236-245.

[8] L. X. Wang and J. M. Mendel, Fuzzy basis functions, universal approximation, and orthogonal least square learning, IEEE Trans, Neural Networks 3(5) (1992), 807-814.

[9] S. Araki, H. Nomura, I. Hayashi and N. Wakami, A fuzzy modeling with iterative generation mechanism of fuzzy inference rules, Journal of Japan Society for Fuzzy Theory and Systems 4(4) (1992), 722-732.

[10] S. Fukumoto, H. Miyajima, K. Kishida and Y. Nagasawa, A destructive learning method of fuzzy inference rules, Proc. of IEEE on Fuzzy Systems, 1995, pp. 687-694.

[11] O. Cordon, A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: designing interpretable genetic fuzzy systems, Int. J. Approx. Reason. 52 (2011), 894-913.

[12] K. Kishida, H. Miyajima, M. Maeda and S. Murashima, A self-tuning method of fuzzy modeling using vector quantization, Proc. of FUZZ-IEEE'97, 1997, pp. 397-402.

[13] S. Fukumoto and H. Miyajima, Learning algorithms with regularization criteria for fuzzy reasoning model, Journal of Innovative Computing, Information and Control 1(1) (2006), 249-263.

[14] S. M. Zhoua and J. Q. Ganb, Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modeling, Fuzzy Sets and Systems 159 (2008), 3091-3131.

[15] J. M. Alonso, L. Magdalena and G. Gonza, Looking for a good fuzzy system interpretability index: an experimental approach, Int. J. Approx. Reason. 51 (2009), 115-134.

[16] J. Yi, N. Yubazaki and K. Hirota, A proposal of SIRMs dynamically connected fuzzy inference model for plural input fuzzy control, Fuzzy Sets and Systems 125 (2002), 79-92.

[17] N. Yubazaki, J. Yi and K. Hirota, SIRMs (single input rule modules) connected fuzzy inference model, J. Advanced Computational Intelligence 1(1) (1997), 23-30.

[18] N. Shigei, H. Miyajima and S. Nagamine, A proposal of fuzzy inference model composed of small-number-of-input rule modules, Proc. of Int. Symp. on Neural Networks: Advances in Neural Networks - Part II, 2009, pp. 118-126.

[19] S. Miike, H. Miyajima, N. Shigei and K. Noo, Fuzzy reasoning model with deletion of rules consisting of small-number-of-input rule modules, Journal of Japan Society for Fuzzy Theory and Intelligent Informatics (2010), 621-629 (in Japanese).

[20] H. Miyajima, N. Shigei and H. Miyajima, Fuzzy inference systems composed of double-input rule modules for obstacle avoidance problems, IAENG Int. J. Comput. Sci. 41(4) (2014), 222-230.

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