Fuzzy Modeling using Vector Quantization based on Input and Output Learning Data

Hirofumi Miyajima, Noritaka Shigei, and Hiromi Miyajima

***Abstract*—Many studies on fuzzy modeling(learning of fuzzy inference systems) with vector quantization(VQ) and steepest descend method (SDM) have been made. It is known that these methods are superior in the number of rules(parameters) compared with other learning methods. Most of conventional learning methods using VQ are ones that determine initial assignment of center parameters for membership functions in antecedent part using only input part of learning data, initial assignment of center parameters for membership functions in antecedent part using all learning data and the initial assignment of all parameters in systems using VQ and the generalized onverse matrix(GIM). These methods are ones that determine the initial assignment of parameters in learning process, and any learning data in learning steps of SDM is selected randomly. On the other hand, it is known that many fuzzy rules are needed at or near the places where output changes rapidly in learning data. Therefore, the rate of output change for learning data must be considered. In this paper, we propose learning methods that any data in learning steps of SDM are selected using the probability based on the rate of output change for learning data. In order to demonstrate the effectiveness of the proposed methods, numerical simulations for function approximation and pattern classification problems are performed.**

***Index Terms*—Fuzzy Inference Systems, Vector Quantization, Neural Gas Network, Steepest Descent Method.**

I. Introduction

MANY studies on fuzzy modeling(learning of fuzzy inference systems) have been made [1], [2]. Their aim is to construct automatically fuzzy inference systems from learning data. Although most of conventional methods are based on steepest descend method(SDM), the obvious drawbacks of them are its large time complexity and getting stuck in a shallow local minimum. Further, there is problems of difficulty dealing with high dimensional spaces [3], [4]. 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, or delete fuzzy rules one by one starting from a sufficiently large number of rules [5], 2) use GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) to determine fuzzy systems [6], 3) use fuzzy inference systems composed of small number of input rule modules, such as SIRMs (Single Input Rule Modules) and DIRMs (Double Input Rule Modules) methods [7], [8], and 4) use a self-organization or a vector quantization technique to determine the initial assignment of learning parameters [5], [9]. Specifically, learning methods using VQ and SDM are superior in the number of rules(parameters) compared with other learning methods [10]. Most of conventional learning methods using VQ are ones that determine initial assignment of parameters for membership functions in antecedent part using only input part of learning data. Therefore, we proposed some learning methods to determine initial assignment of center parameters for membership functions in antecedent part using all learning data. Further, we proposed learning methods determining the initial assignment of weight parameters in consequent part [12], [13]. These methods are ones that determine the initial assignment of learning parameters in learning process, and any learning data is selected randomly in learning steps of SDM [12]–[14]. On the other hand, it is known that many rules are needed at or near the places where output changes rapidly in learning data. Therefore, the rate of change for output data must be considered. Little learning methods selected any data using the probability based on the rate of output change for learning data have been proposed. In this paper, we propose learning methods that any data are selected using the probability based on the rate of output change for learning data in learning process of SDM. In order to demonstrate the effectiveness of the proposed method, numerical simulations for function approximation and pattern classification problems are performed.

Hirofumi Miyajima is with the Graduate School of Biomedical Sciences, Nagasaki Univercity, Sakamoto, Nagasaki, Japan e-mail: k3768085@kadai.jp.

Noritaka Shigei is with Kagoshima University, e-mail: shigei@eee.kagoshima-u.ac.jp

Hiromi Miyajima is with Kagoshima University, e-mail: miya@eee.kagoshima-u.ac.jp.

II. Preliminaries

A. The conventional fuzzy inference model

The conventional fuzzy inference model using SDM is described [1], [2]. Let $Z_j = \{1, \cdots, j\}$ and $Z_j^* = \{0, 1, \cdots, j\}$ for the positive integer j. Let R be the set of real numbers. Let $\boldsymbol{x} = (x_1, \cdots, x_m)$ and y^r be input and output data, respectively, where $x_i \in R$ for $i \in Z_m$ and $y^r \in R$. Then the rule of simplified fuzzy inference model is expressed as

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

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.

A membership value of the antecedent part μ_j for input $\boldsymbol{x}$ is expressed as

$$\mu_i = \prod_{j=1}^{m} M_{ij}(x_j). \tag{2}$$

If Gaussian membership function is used, then M_{ij} is expressed as follow:

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

, where c_{ij} and b_{ij} are the center and the width values of M_{ij}, respectively. The inference output y^* is calculated by

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Eq.(4).

$$y^* = \frac{\sum_{i=1}^{n} \mu_i \cdot w_i}{\sum_{i=1}^{n} \mu_i}. \quad (4)$$

In order to construct the effective model, the conventional learning is introduced. The objective function E is determined to evaluate the inference error between the desirable output y^r and the inference output y^*.

In this section, we describe the conventional learning algorithm [2].

Let $D = \{(x_1^p, \cdots, x_m^p, y^p) | p \in Z_P\}$ and $D^* = \{(x_1^p, \cdots, x_m^p) | p \in Z_P\}$ be the set of learning data and the set of input data of D, respectively. The objective of learning is to minimize the following mean square error(MSE):

$$E = \frac{1}{P} \sum_{p=1}^{P} (y_p^* - y^p)^2. \quad (5)$$

, where y_p^* is the inference output for the data $\boldsymbol{x}^p$.

In order to minimize the objective function E, each parameter $\alpha \in \{c_{ij}, b_{ij}, w_j\}$ is updated based on SDM as follows [1], [2]:

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

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

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

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

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

The conventional learning algorithm is shown as Fig.1 [1], [2], [11], where n_0, θ and T_{max} are the initial number of rules, threshold and the maximum number of learning, respectively. Note that the method is generative one. The method is called learning algorithm A.

B. Neural gas and K-means methods

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

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

$$i(\boldsymbol{v}) = \arg\min_{i \in Z_r} ||\boldsymbol{v} - \boldsymbol{c}_i|| \quad (10)$$

, where $||\boldsymbol{a} - \boldsymbol{b}||$ means the distance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

From the finite set C, V is partioned as follows:

$$V_i = \{\boldsymbol{v} \in V | ||\boldsymbol{v} - \boldsymbol{c}_i|| \leq ||\boldsymbol{v} - \boldsymbol{c}_j|| \ for \ j \in Z_r\} \quad (11)$$

The evaluation function for the partition is defined as follows:

$$E = \sum_{i=1}^{r} \sum_{\boldsymbol{v} \in V_i} ||\boldsymbol{v} - \boldsymbol{c}_{i(\boldsymbol{v})}||^2 \quad (12)$$

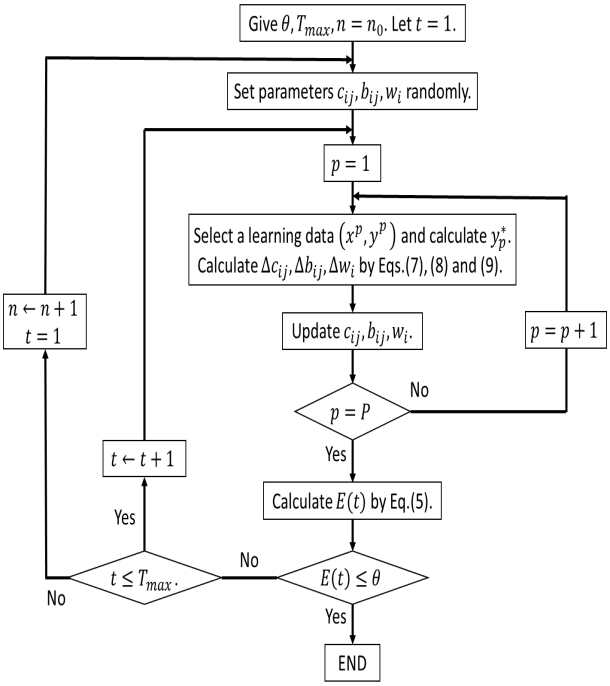


Fig. 1. The flowchart of the conventional learning algorithm

For neural gas method [15], the following method is used:

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

$$||\boldsymbol{v} - \boldsymbol{c}_j|| < ||\boldsymbol{v} - \boldsymbol{c}_{i_k}|| \quad (13)$$

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

$$\triangle \boldsymbol{c}_i = \varepsilon \cdot h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{c})) \cdot (\boldsymbol{v} - \boldsymbol{c}_i) \quad (14)$$

$$h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{c})) = \exp(-k_i(\boldsymbol{v}, \boldsymbol{c})/\lambda) \quad (15)$$

where $\varepsilon \in [0, 1]$ and $\lambda > 0$. The number λ is called decay constant.

If $\lambda \to 0$, Eq.(14) becomes equivalent to the K-means method [15]. Otherwise, not only the winner $\boldsymbol{c}_{i_0}$ but the second, third nearest reference vector $\boldsymbol{c}_{i_1}$, $\boldsymbol{c}_{i_2}$, etc., are also updated.

Let $p(\boldsymbol{v})$ be the probability distribution of data vectors for V. The flowchart of the conventional neural gas algorithm is shown as Fig.2 [15], where ε_{int}, ε_{fin}, θ and T_{max} are learning constants, threshold and the maximum number of learning, respectively. The method is called learning algorithm NG.

If the data distribution $p(\boldsymbol{v})$ is not given in advance, a stochastic sequence of input data $\boldsymbol{v}(1), \boldsymbol{v}(2), \cdots$ which is based on $p(\boldsymbol{v})$ is given [15].

By using Learning Algorithm NG, learning method of fuzzy systems is shown as follows [9], [10] : In this case, assume that the distribution of learning data $\boldsymbol{D}^*$ is discrete uniform one. Let n_0 be the initial number of rules.

Learning Algorithm B

Step B1 : For learning data D^*, Learning Algorithm NG is performed by using D^* as the set V. The set D^* is encoded by the set C of reference vectors, where $|C| = n_0$.

Step B2 : The set of center parameters of fuzzy rules is set

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

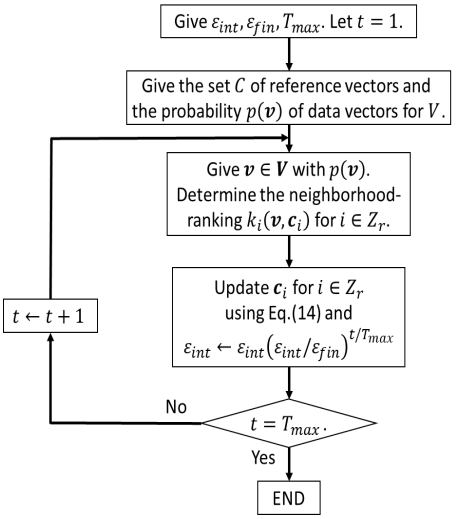


Fig. 2. Neural Gas method

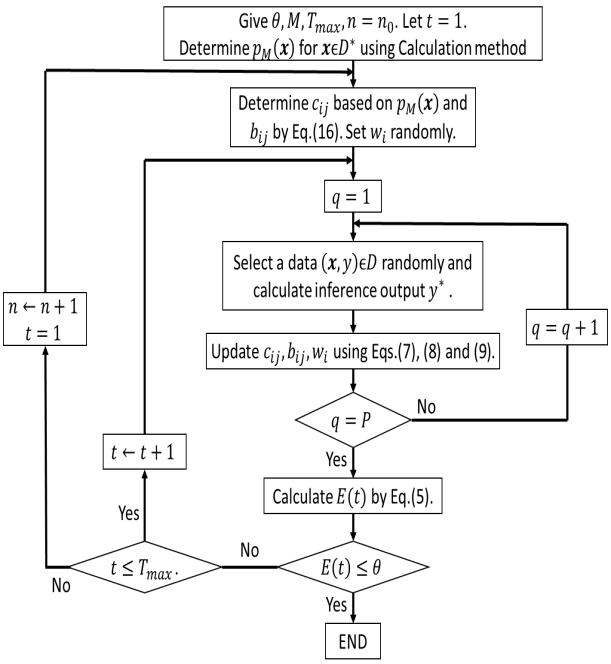


Fig. 3. The flowchart of learning algorithm C

to the set C.
Let

$$b_{ij} = \frac{1}{m_i} \sum_{\boldsymbol{x}_k \in C_i} (c_{ij} - x_{kj})^2, \tag{16}$$

where C_i and m_i are the i-th cluster for C and the number of learning data for $i \in Z_{n_0}$. Each initial weight w_i is selected randomly.

Step B3 : Learning algorithm A for initial parameters c_{ij}, b_{ij} and w_i are performed.

C. The probability based on the rate of output change for learning data

Learning Algorithm B is a method that determines the initial assignment of fuzzy rules by vector quantization using the set D^* of input for learning data. In this case, the set of output in learning data D is not used to determine the initial assignment of fuzzy rules. In the previous paper, we proposed a method considering both input and output data to determine the initial assignment of fuzzy rules [5].

Based on the literature [5], the probability distribution for D^* is defined as follows : Let D and D^* be the sets of learning data defined in 2.1.

Calculation of the probability for learning data

Step 1 : Give an input data $\boldsymbol{x}^i \in D^*$, we determine the neighborhood-ranking $(\boldsymbol{x}^{i_0}, \boldsymbol{x}^{i_1}, \cdots, \boldsymbol{x}^{i_k}, \cdots, \boldsymbol{x}^{i_{P-1}})$ of the vector $\boldsymbol{x}^i$ with $\boldsymbol{x}^{i_0} = \boldsymbol{x}^i$, $\boldsymbol{x}^{i_1}$ being closest to $\boldsymbol{x}^i$ and $\boldsymbol{x}^{i_k} (k = 0, \cdots, P-1)$ being the vector $\boldsymbol{x}^i$ for which there are k vectors $\boldsymbol{x}^j$ with $||\boldsymbol{x}^i - \boldsymbol{x}^j|| < ||\boldsymbol{x}^i - \boldsymbol{x}^{i_k}||$.

Step 2 : Determine $H(\boldsymbol{x}^i)$ which shows the rate of change of inclination of the output around output data to input data $\boldsymbol{x}^i$, by the following equation:

$$H(\boldsymbol{x}^i) = \sum_{l=1}^{M} \frac{|y^i - y^{i_l}|}{||\boldsymbol{x}^i - \boldsymbol{x}^{i_l}||} \tag{17}$$

, where $\boldsymbol{x}^{i_l}$ for $l \in Z_M$ means the l-th neighborhood-ranking of $\boldsymbol{x}^i$, $i \in Z_P$ and y^i and y^{i_l} are output for input $\boldsymbol{x}^i$ and $\boldsymbol{x}^{i_l}$, respectively. The number M means the range of ranking considering $H(\boldsymbol{x})$.

Step 3 : Determine the probability $p_M(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i \in D^*$ by normalizing $H(\boldsymbol{x}^i)$.

$$p_M(\boldsymbol{x}^i) = \frac{H(\boldsymbol{x}^i)}{\sum_{j=1}^{P} H(\boldsymbol{x}^j)} \tag{18}$$

and $\sum_{i=1}^{P} p_M(\boldsymbol{x}^i) = 1$.

[Example 1]

Let us explain how to compute $p_M(\boldsymbol{x})$ using $y = sin(\pi x_1^3) x_2$ as shown in Fig.4, where $x_1, x_2, y \in [0, 1]$.

Assume that four learning data are given as follows :

$\boldsymbol{x}$	y
$\boldsymbol{x}^1 = (0.2, 0.2)$	0.005
$\boldsymbol{x}^2 = (0.2, 0.8)$	0.020
$\boldsymbol{x}^3 = (0.8, 0.2)$	0.200
$\boldsymbol{x}^4 = (0.8, 0.8)$	0.799

Let $M = 2$.

Then, $H(\boldsymbol{x}^1)$ is calculated as follows :

$$\begin{aligned} H(\boldsymbol{x}^1) &= \frac{|y^1 - y^2|}{||\boldsymbol{x}^1 - \boldsymbol{x}^2||} + \frac{|y^1 - y^3|}{||\boldsymbol{x}^1 - \boldsymbol{x}^3||} \\ &= 0.35 \end{aligned} \tag{19}$$

because the first and second closest vectors for $\boldsymbol{x}^1$ are $\boldsymbol{x}^2$ and $\boldsymbol{x}^3$.

Likewise, we obtained $H(\boldsymbol{x}^2) = 1.325$, $H(\boldsymbol{x}^3) = 1.325$ and $H(\boldsymbol{x}^4) = 2.3$. From the Eq.(18), each of $p_2(\boldsymbol{x})$'s is calculated as $p_2(\boldsymbol{x}^1) = 0.066$, $p_2(\boldsymbol{x}^2) = 0.25$, $p_2(\boldsymbol{x}^3) = 0.25$, $p_2(\boldsymbol{x}^4) = 0.434$.

If $M = 3$, then the following result is obtained : $p_3(\boldsymbol{x}^1) = 0.143$, $p_3(\boldsymbol{x}^2) = 0.21$, $p_3(\boldsymbol{x}^3) = 0.169$, $p_3(\boldsymbol{x}^4) = 0.477$.

In both cases, the rate of output change for $\boldsymbol{x}^4 = (0.8, 0.8)$ is large compared to other points, so $p_M(\boldsymbol{x}^4)$ is large.

The flowchart of learning algorithm C using $p_M(\boldsymbol{x})$ is shown in Fig.3 [5].

Then, let us show how the assignment of fuzzy rules changes after learning using an example.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

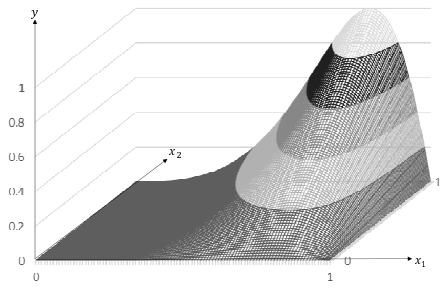


Fig. 4. The figure of $y = sin(\pi x_1^3)x_2$

[Example 2]

Let us consider two cases with $M = 1$ and 100 using $y = sin(\pi x_1^3)x_2$, where Fig.5(a) and (b) are the initial assignment of center parameters for four fuzzy rules using $p_M(\boldsymbol{x})$ with $M = 1$ and 100 for the number of learning data $P = 500$, respectively. Fig.5(c) and (d) are the assignment of fuzzy rules after learning for $M = 1$ and 100, respectively. The result shows that the case of $M = 100$ is superior in the assignment of fuzzy rules after learning to the case of $M = 1$. Please see Ref. [11] about the detailed explanation of Algorithm C.

It is known that many rules are needed at or near palaces where output changes quickly for learning data. The probability $p_M(\boldsymbol{x})$ is a technique to find the optimum places for the assignment of fuzzy rules. The algorithm C is a heuristic method to find the optimum number of M.

III. The proposed method

It is shown that learning algorithms A, B and C using VQ and SDM is effective in accuracy and the number of rules to other methods. Most of conventional learning methods using VQ are ones that determine initial assignment of parameters in antecedent part for membership functions. However, little learning methods using VQ in learning process of SDM have been proposed.

The method using VQ in SDM means that each learning data in SDM is not selected randomly, but selected based on $p_M(\boldsymbol{x})$. Therefore, each data existing the place where output rapidly changes is more likely to be selected. Let explain it using Example 1.

[Example 3]

Let $M = 2$ in Example 1. Then $p_2(\boldsymbol{x}^1) = 0.066$, $p_2(\boldsymbol{x}^2) = 0.25$, $p_2(\boldsymbol{x}^3) = 0.25$ and $p_2(\boldsymbol{x}^4) = 0.434$, so learning data $(\boldsymbol{x}^1, y^1)$, $(\boldsymbol{x}^2, y^2)$, $(\boldsymbol{x}^3, y^3)$ and $(\boldsymbol{x}^4, y^4) \in D$ are selected with the probability 0.066, 0.25, 0.25 and 0.434, respectively.

In this case, output change for $(\boldsymbol{x}^4, y^4)$ and $(\boldsymbol{x}^1, y^1)$ are rapidly and flat as shown in Fig.4, respectively. It seems to assign a lot of fuzzy rules at or near the place $(\boldsymbol{x}^4, y^4)$.

In this section, we propose three algorithms using $p_M(\boldsymbol{x})$ in learning steps of SDM corresponding to algorithms A, B and C. They are called algorithms(methods) A', B' and C'. The algorithm C' is only shown as follows:

Learning Algorithm C'

Step 1 : θ, T_{max}^0, T_{max} and M are set. Initial values of c_{ij}, b_{ij} and w_i are set randomly. $n \leftarrow n_0$.
Step 2 : Let $t = 1$.
Step 3 : Select a data $(\boldsymbol{x}^p, y^p)$ based on $p_M(\boldsymbol{x}^p)$ for $p \in Z_P$.
Step 4 : Update c_{ij} by Eq.(14).
Step 5 : If $t < T_{max}^0$, go to Step 3 with $t \leftarrow t+1$, otherwise go to Step 6 with $t \leftarrow 1$.
Step 6 : Determine b_{ij} by Eq.(16).
Step 7 : Let $q = 1$.
Step 8 : Given a data $(\boldsymbol{x}, y)$ based on $p_M(\boldsymbol{x})$ for $\boldsymbol{x} \in D$.
Step 9 : Calculate μ_i and y^* by Eqs.(2) and (4).
Step 10 : Update parameters c_{ij}, b_{ij} and w_{ij} by Eqs.(7), (8) and (9).
Step 11 : If $q < P$ then go to Step 8 with $q \leftarrow q+1$.
Step 12 : If $E < \theta$ or $t > T_{max}$ go to Step 13, otherwise go to Step 8 with $t \leftarrow t+1$, where E is computed as Eq.(5).
Step 13 : If $E < \theta$ the algorithm terminate, otherwise go to Step 2 with $n \leftarrow n+1$ and c_{ij}, b_{ij} and w_i are set randomly.

The feature of the proposed method is that learning data are selected from the probability $p_M(\boldsymbol{x})$ in both determining of the initial assignment of parameters and learning steps of SDM(See steps 3 and 8 in learning algorithm C').

In order to compare the proposed method with conventional ones, the following methods are introduced (See Fig.6):

(A) Method A is one based on the conventional algorithm of Fig.1 [1], [2]. Initial parameters of $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are set randomly and all parameters are updated by SDM using learning data selected randomly until the inference error become sufficiently small.

(B) Method B is known as learning method of RBF networks [2], [3], [10]. Initial values of $\boldsymbol{c}$ are determined using D^* by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters $\boldsymbol{w}$ are randomly selected. Further, all parameters are updated by SDM until the inference error become sufficiently small.

(C) Method C was introduced in the chapter II. Initial values of $\boldsymbol{c}$ are determined using D by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters are randomly selected. Further, all parameters are updated by SDM until the inference error become sufficiently small.

(A') Method A' is the proposed one. Initial parameters of $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are set randomly and all parameters are updated using SDM using learning data based on $p_M(\boldsymbol{x})$ until the inference error become sufficiently small.

(B') Method B' is the proposed one. Initial values of $\boldsymbol{c}$ are determined using D^* by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters $\boldsymbol{w}$ are randomly selected. Further, all parameters are updated by SDM based on $p_M(\boldsymbol{x})$ until the inference error become sufficiently small.

(C') Method C' is the proposed one. Initial values of $\boldsymbol{c}$ are determined using D by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters are randomly selected. Further, all parameters are updated by SDM based on $p_M(\boldsymbol{x})$ until the inference error become sufficiently small.

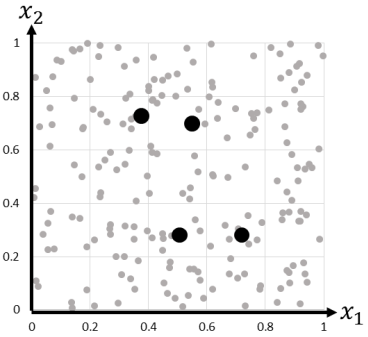
IV. Numerical simulations

In order to show the effectiveness of proposed algorithms, simulations of function approximation and classification problems are performed.

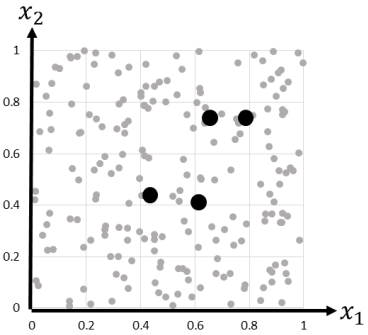
A. Function approximation

The systems are identified by fuzzy inference systems. This simulation uses four systems specified by the following functions with 4-dimensional input space $[0, 1]^4$(Eqs.(20) and

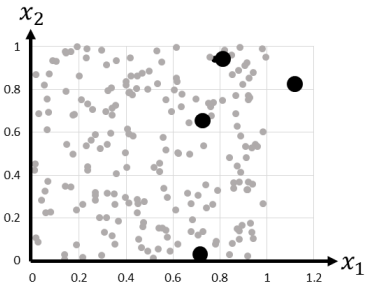
ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



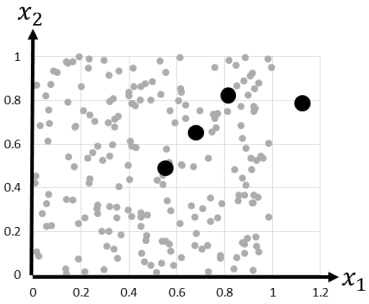
(a)The initial assignment of rules for $M = 1$



(b)The initial assignment of rules for $M = 100$



(c)The assignment of rules after learning for $M = 1$



(d)The assignment of rules after learning for $M = 100$

Fig. 5. The figures (a) and (b) show the initial assignment for $M = 1$ and 100, respectively, where ○ and ● mean the places of learning data and center parameters of fuzzy rules. The figures (c) and (d) show the assignment after learning $M = 1$ and 100, respectively.

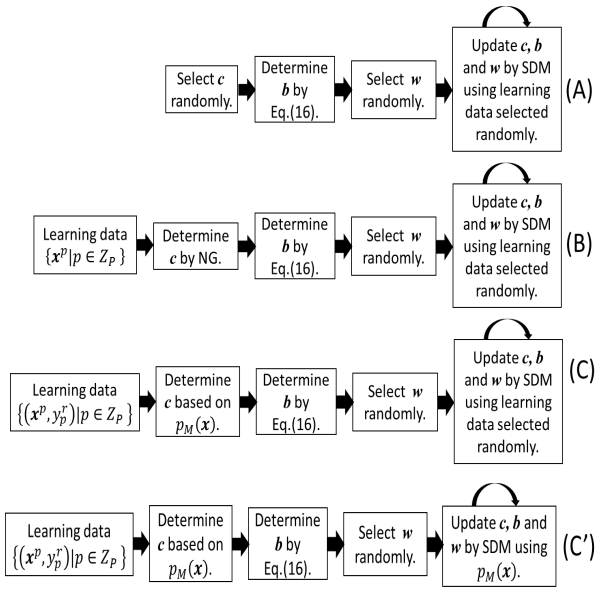


Fig. 6. Concept of conventional and proposed methods, where SDM andd NG mean Steepest Descent Method and Neural Gas method. The algorithm C' is only shown, and algorithms A' and B' are also defined in the same way.

(21)) and $[-1, 1]^4$((22) and (23)), and one output with the range $[0, 1]$;

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6)}{12} \quad (20)$$

$$y = \frac{(\sin(2\pi x_1) \times \cos(x_2) \times \sin(\pi x_3) \times x_4 + 1.0)}{2.0} \quad (21)$$

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

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6)}{446.52} \quad (23)$$

TABLE I
THE RESULTS FOR FUNCTION APPROXIMATION

		Eq(20)	Eq(21)	Eq(22)	Eq(23)
A	the number of rules	4.2	13.6	7.2	5.1
	MSE for Learning($\times 10^{-4}$)	0.40	0.71	0.43	0.28
	MSE of Test($\times 10^{-4}$)	0.52	1.09	1.00	0.49
B	the number of rules	5.6	14.9	5.2	3.7
	MSE of Learning($\times 10^{-4}$)	0.18	0.77	0.49	0.33
	MSE of Test($\times 10^{-4}$)	0.27	1.42	1.11	0.48
C	the number of rules	4.8	15.6	5.5	4.0
	MSE of Learning($\times 10^{-4}$)	0.21	0.72	0.54	0.88
	MSE of Test($\times 10^{-4}$)	0.34	1.33	0.69	0.53
A'	the number of rules	3.3	7.8	5.5	3.7
	MSE for Learning($\times 10^{-4}$)	0.22	0.63	0.43	0.27
	MSE of Test($\times 10^{-4}$)	0.30	1.10	0.58	0.33
B'	the number of rules	3.1	7.0	4.9	3.4
	MSE of Learning($\times 10^{-4}$)	0.21	0.66	0.57	0.25
	MSE of Test($\times 10^{-4}$)	0.26	2.30	0.69	0.27
C'	the number of rules	3.2	7.1	4.7	3.1
	MSE of Learning($\times 10^{-4}$)	0.21	0.65	0.60	0.22
	MSE of Test($\times 10^{-4}$)	0.26	0.91	0.73	0.25

As the initial conditions of simulations, $T_{max} = 50000$, $K_c = 0.01$, $K_b = 0.01$, $K_c = 0.1$, $\varepsilon_{init} = 0.1$, $\varepsilon_{fin} = 0.01$, $\lambda = 0.7$, $\theta = 1.0\times10^{-4}$ and $M = 200$ are used. The numbers of learning and test data are 512 and 6400, respectively.

Table I shows the result of simulation, where the number of rules, MSE's for learning and test data are shown. In Table I, the number of rules means one when the threshold $\theta = 1.0\times10^{-4}$ of inference error is achieved in learning.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE II
THE DATASET FOR PATTERN CLASSIFICATION

	Iris	Wine	BCW
The number of data	150	178	683
The number of input	4	13	9
The number of class	3	3	2

TABLE III
THE RESULT FOR PATTERN CLASSIFICATION

		Iris	Wine	BCW
A	the number of rules	3.4	7.8	14.4
	RM for Learning(%)	3.0	1.4	1.6
	RM of Test(%)	3.3	10.3	4.3
B	the number of rules	2.0	20.8	26.0
	RM of Learning(%)	3.3	13.6	2.2
	RM of Test(%)	3.3	16.6	3.5
C	the number of rules	3.4	7.4	9.6
	RM of Learning(%)	2.8	2.1	2.0
	RM of Test(%)	4.7	5.1	4.6
A'	the number of rules	2.2	2.2	3.7
	RM for Learning(%)	2.7	1.5	1.5
	RM of Test(%)	3.6	7.7	4.1
B'	the number of rules	2.2	2.6	2.5
	RM of Learning(%)	2.4	1.3	1.7
	RM of Test(%)	3.9	7.7	4.0
C'	the number of rules	2.2	2.5	2.6
	RM of Learning(%)	2.4	1.4	1.6
	RM of Test(%)	3.8	8.2	4.0

The result of simulation is the average value from twenty trials. As a result, proposed methods A', B' and C' reduce the number of rules compared to conventional methods. Specifically, algorithm C' is superior to other algorithms.

B. Classification problems for UCI database

Iris, Wine and BCW data from UCI database shown in Table II are used as the second numerical simulation [16]. In this simulation, 5-fold cross-validation is used. As the initial conditions for classification problem, $T_{max} = 50000$, $K_c = 0.001$, $K_b = 0.001$, $K_w = 0.05$, $\varepsilon_{init} = 0.1$, $\varepsilon_{fin} = 0.01$ and $\lambda = 0.7$ are used. Further, $M = 100$ and $\theta = 1.0\times10^{-2}$ for Iris and Wine and $M = 200$ and $\theta = 2.0\times10^{-2}$ for BCW are used.

Table III shows the result of classification problem. In Table III, the number of rules, RM's for learning and test data are shown, where RM means the rate of misclassification. As a result, it is shown that proposed methods A', B' and C' reduce the number of rules in classification problem.

Let us consider the reason why we can get the good result by using the probability $p_M(\boldsymbol{x})$. In the conventional learning method, parameters are updated by any data selected randomly from the set of learning data. In the proposed method, parameters are updated by any data selected from the probability $p_M(\boldsymbol{x})$. The function $p_M(\boldsymbol{x})$ is determined based on output change for the set of learning data, so many fuzzy rules are likely to generate at or near the places where output change is large for the set of learning data. For example, if the number of learning time is 100 and $p_M(\boldsymbol{x}^0) = 0.5$, then learning data $\boldsymbol{x}^0$ is selected 50 times from the set of learning data in learning. As a result, membership functions are likely to generate at or near the places where output change is large for the set of learning data. The probability $p_M(\boldsymbol{x})$ is considered as a method to improve the local search of SDM.

V. CONCLUSION

In this paper, we propose learning methods that any data in learning steps of SDM is selected based on a probability based on the rate of output change for learning data. It means that each learning data in SDM is not selected randomly, but selected based on the probability for output change. Therefore, each data existing the place where output rapidly changes is more likely to be selected. In order to demonstrate the effectiveness of the proposed method, numerical simulations for function approximation and pattern classification problems were performed. In the future work, we will propose more effective learning algorithm using the probability of output change for learning data compared to other methods and consider to apply the proposed method to learning of neural network.

REFERENCES

[1] B. Kosko, Neural Networks and Fuzzy Systems, A Dynamical Systems Approach to Machine Intelligence, Prentice Hall, Englewood Cliffs, NJ, 1992.
[2] M.M. Gupta, L. Jin and N. Homma, Static and Dynamic Neural Networks, IEEE Press, 2003.
[3] 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.
[4] 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, pp.3091-3131, 2008.
[5] K. Kishida and H. Miyajima, A Learning Method of Fuzzy Inference Rules using Vector Quantization, Proc. of the Int. Conf. on Artificial Neural Networks, Vol.2, pp.827-832, 1998.
[6] O. Cordon, A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems, Designing interpretable genetic fuzzy systems, Journal of Approximate Reasoning, 52, pp.894-913, 2011.
[7] N. Yubazaki, J. Yi and K. Hirota, SIRMS(Single Input Rule Modules) Connected Fuzzy Inference Model, J. Advanced Computational Intelligence, 1, 1, pp.23-30, 1997.
[8] H. Miyajima, N. Shigei and H. Miyajima, Fuzzy Inference Systems Composed of Double-Input Rule Modules for Obstacle Avoidance Problems, IAENG International Hournal of Computer Science, Vol. 41, Issue 4, pp.222-230, 2014.
[9] K. Kishida, H. Miya jima, M. Maeda and S. Murashima, A Self-tuning Method of Fuzzy Modeling using Vector Quantization, Proceedings of FUZZ-IEEE'97, pp397-402, 1997.
[10] S. Fukumoto, H. Miyajima, N. Shigei and K. Uchikoba, A Decision Procedure of the Initial Values of Fuzzy Inference System Using Counterpropagation Networks, Journal of Signal Processing, Vol.9, No.4, pp.335-342, 2005.
[11] H. Miyajima, N. Shigei, K. Kishida, Y. Akiyoshi and H. Miyajima, An Improved Learning Algorithm of Fuzzy Inference Systems using Vector Quantization, Advanced in Fuzzy Sets and Systems, Vol.20, pp.155-175, 2015.
[12] H. Miyajima, N. Shigei, H. Miyajima, Fast Learning Algorithm for Fuzzy Inference Systems Using Vector Quantization, Int. MultiConference of Engineers and Computer Scientists 2016, Vol.I, pp.1-6, Hong Kong, March, 2016.
[13] H. Miyajima, N. Shigei and H. Miyajima, The Ability of Learning Algorithms for Fuzzy Inference Systems using Vector Quantization, ICONIP 2016, LNCS 9950, pp.479, Kyoto, October, 2016.
[14] W. Pedrycz, H. Izakian, Cluster-Centric Fuzzy Modeling, IEEE Trans. on Fuzzy Systems, Vol. 22, Issue 6, pp. 1585-1597, 2014.
[15] T. M. Martinetz, S. G. Berkovich and K. J. Schulten, Neural Gas Network for Vector Quantization and its Application to Time-series Prediction, IEEE Trans. Neural Network, 4, 4, pp.558-569, 1993.
[16] UCI Repository of Machine Learning Databases and Domain Theories, ftp://ftp.ics.uci.edu/pub/machinelearning-Databases.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)