Fuzzy Modeling using Vector Quantization and Local Linear Mapping

Hirofumi Miyajima, Noritaka Shigei, Hiromi Miyajima

Abstract—**Fuzzy modeling has been extensively studied. It has been shown that fuzzy modeling using vector quantization (VQ) and steepest descent method (SDM) is effective in terms of the number of rules (parameters). In the methods, the initial parameters of fuzzy rules by using VQ with learning data firstly are determined, then the parameters are adjusted by using SDM. On the other hand, Neural Gas (NG) is known as a novel approach of VQ and NG with local linear mapping (LLM) has been applied to a time-series prediction problem. In the application, the predicted value in each of subregions is approximated by using a corresponding linear mapping. It has been demonstrated that, compared with RBF, NG with LLM is advantageous in terms of the accuracy and the number of rules. The idea of NG with LLM has been applied to fuzzy modeling with TS fuzzy model, and its effectiveness has been demonstrated. However, the effectiveness of this approach has not been been confirmed for simpler fuzzy model such as simplified fuzzy model. In this paper, the method using NG with LLM to fuzzy modeling with simplified fuzzy inference model is proposed and the similar method using k-means instead of NG is also proposed. In the proposed methods, the initial parameters of fuzzy rules including the weights in the consequent part by using NG (k-means) are firstly defined, and then the parameters by using SDM are adjusted. The effectiveness of the proposed methods with simplified fuzzy modeling is demonstrated in numerical simulations of function approximation and classification problems.**

Index Terms—**Simplified Fuzzy Inference Model, Vector Quantization, Neural Gas, k-means, Local Linear Mapping**

I. Introduction

WITH increasing interest in Artificial Intelligence (AI), extensive studies related to Machine Learning (ML) have been done. In the field of ML, supervised learning such as backpropagation, unsupervised one such as K-means, and Reinforcement Learning (RL) are well known. ML for fuzzy inference models, called fuzzy modeling, is one of the supervised ones, and extensive studies on fuzzy modeling have also been made [1], [2]. Although most of fuzzy modeling methods are based on steepest descent method (SDM), their obvious drawbacks are its large time complexity, getting stuck easily in shallow local minima and difficulty of applying to high dimensional data. In order to overcome the drawbacks, the following has been developed: 1) creating fuzzy rules one by one starting from a few rules, or deleting fuzzy rules one by one starting from many rules [3], 2) using evolutionary algorithms such as GA and PSO to determine parameters of fuzzy inference systems [4], 3) fuzzy inference systems composed of rule modules with a few inputs such as SIRMs (Single Input Rule Modules) and DIRMs (Double Input Rule Modules) methods [5], and 4) determining the initial assignment of learning parameters by using self-organization map (SOM) or VQ [6], [7]. Especially, with fuzzy modeling using VQ methods such as k-means, Neural-Gas (NG), SOM and fuzzy c-means, there are many studies on how to realize the system with high accuracy and a few rules [8], [9]. In the methods, the initial parameters of the antecedent part of rules by VQ using only input or both input and output of learning data are firstly determined, and then all the parameters by using SDM and adjusted. Further, in addition to initial assignments of the antecedent part, it also has been proposed to determine the initial assignment of weight parameters of the consequent part by using VQ [8], [9].

Neural Gas (NG) is known as a novel approach of VQ and NG with local linear mapping (LLM) has been applied to a time-series prediction problem [10]. In the application, the predicted value in each of subregions is approximated by using a corresponding linear mapping. It has been demonstrated that, compared with k-means with RBF, NG with LLM is advantageous in terms of the number of subregions, each of which is approximated by using a linear mapping. The similar idea has been applied to fuzzy modeling with TS fuzzy model, and its effectiveness has been demonstrated [11]. However, the effectiveness of the similar approach has not been been confirmed for simpler fuzzy model such as simplified fuzzy modeling [12], [14].

In this paper, the concept of Vector Quantization such as NG and k-means with LLM is applied to fuzzy modeling with simplified fuzzy inference model. Before adjusting the parameters by using SDM, the initial parameters of fuzzy rules including the weights in the consequent part are determined by using NG (k-means) and supervised learning in the proposed methods. The determination of the initial parameters of the consequent part firstly divides the input space into Voronoi subregions, and then determines a constant value well approximating the output value of each subregion by using supervised learning. The effectiveness of the proposed method with simplified fuzzy modeling is demonstrated in numerical simulations of function approximation and classification problems. In Section II, the conventional learning methods of fuzzy model, NG, k-means and LLM are introduced. In Section III, a learning method for the simplified fuzzy models using NG and k-means are proposed. In Section IV, numerical simulations for function approximation and classification problems are performed to demonstrate the effectiveness of the proposed methods.

Hirofumi Miyajima is Shiool of Information and Data Sciences, Nagasaki University, 1-14, Bunkyomachi, Nagasaki city, Nagasaki, Japan e-mail: miyajima@nagasaki-u.ac.jp

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

Hiromi Miyajima is with Former Kagoshima University, email: k2356323@kadai.jp

II. Preliminaries

A. The simplified fuzzy inference models

The conventional (simplified) fuzzy inference model using SDM is described [1]. 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 fuzzy inference model is expressed as

$$R_i \ : \text{ if } x_1 \text{ is } M_{i1} \text{ and} \cdots x_j \text{ is } M_{ij} \cdots \text{ and } x_m \text{ is } M_{im} \text{ then } y \text{ is } w_i, \quad (1)$$

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

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

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

The output y^* of fuzzy inference is obtained as follows:

$$y^* = \frac{\sum_{i=1}^{n} \mu_i w_i}{\sum_{i=1}^{n} \mu_i} \quad (3)$$

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

$$M_{ij}(x_j) = \exp\left(-\frac{1}{2} \left(\frac{x_j - c_{ij}}{b_{ij}} \right)^2 \right) \quad (4)$$

where c_{ij} and b_{ij} denote the center and the width values of M_{ij}, respectively.

In order to realize the effective model, conventional learning is introduced.

Let $\boldsymbol{D} = \{(x_1^p, \cdots, x_m^p, y_p^r) | p \in Z_P\}$ be the set of learning data. The objective of learning is to minimize the following mean square error (MSE). The objective function E is determined to evaluate the inference error between the desirable output y^r and the inference output y^* as follows :

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

In order to minimize the objective function E, each parameter $\alpha \in \{c_{ij}, b_{ij}, w_i\}$ is updated based on SDM (Steepest Descent Method) as follows [1]:

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

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.

$$\frac{\partial E}{\partial w_i} = \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} (y^* - y^r) \quad (7)$$

$$\frac{\partial E}{\partial c_{ij}} = \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} (y^* - y^r)(w_i - y^*) \frac{x_j - c_{ij}}{b_{ij}^2} \quad (8)$$

$$\frac{\partial E}{\partial b_{ij}} = \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} (y^* - y^r)(w_i - y^*) \frac{(x_j - c_{ij})^2}{b_{ij}^3} \quad (9)$$

where $i \in Z_n$ and $j \in Z_m$.

The conventional learning algorithm is shown as follows [1], where θ and T_{max} are the threshold and the maximum number of learning, respectively. Note that the method is a generative one, which creates fuzzy rules one by one starting from any number of rules. The method is called learning algorithm A.

Learning Algorithm A

Input : Learning data $\boldsymbol{D} = \{(x_1^p, \cdots, x_m^p, y_p^r) | p \in Z_P\}$
Output : Parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$

Step A1 : The initial number n of rules is set to n_0. Let $t = 1$.
Step A2 : The parameters c_{ij}, b_{ij} and w_i are set randomly.
Step A3 : Let $p = 1$.
Step A4 : A data $(x_1^p, \cdots, x_m^p, y_p^r) \in \boldsymbol{D}$ is given.
Step A5 : From Eqs.(2) and (3), μ_i and y^* are computed, respectively.
Step A6 : Parameters w_i, c_{ij} and b_{ij} are updated by Eqs.(7), (8) and (9), respectively.
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 Eq.(5). If $E(t) > \theta$ and $t < T_{max}$, then go to Step A3 with $t \leftarrow t + 1$ else if $E(t) \leq \theta$, 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 $t = 1$.

B. Neural gas and K-means

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.

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

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

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

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

For NG [10], the following method is used:

Given an input data 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

$$||\boldsymbol{v} - \boldsymbol{u}_j|| < ||\boldsymbol{v} - \boldsymbol{u}_{i_k}|| \quad (12)$$

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

$$\begin{aligned} \triangle \boldsymbol{u}_i &= \varepsilon h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i))(\boldsymbol{v} - \boldsymbol{u}_i) && (13) \\ h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i)) &= \exp\left(-k_i(\boldsymbol{v}, \boldsymbol{u}_i)/\lambda\right)) && (14) \\ \varepsilon &= \varepsilon_{int} \left(\frac{\varepsilon_{fin}}{\varepsilon_{int}} \right)^{\frac{t}{T_{max}}} \end{aligned}$$

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

The evaluation function for the partition is defined as follows:

$$E = \sum_{\boldsymbol{u}_i \in U} \sum_{\boldsymbol{v} \in V} \frac{h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i))}{\sum_{\boldsymbol{u}_l \in U} h_\lambda(k_l(\boldsymbol{v}, \boldsymbol{u}_l))} ||\boldsymbol{v} - \boldsymbol{u}_{i(\boldsymbol{v})}||^2 \quad (15)$$

If $\lambda \to 0$, Eq.(13) becomes equivalent to the K-means [10].

In k-means, only the winner $\boldsymbol{u}_{i_0}$ is updated in learning. In NG, not only the winner $\boldsymbol{u}_{i_0}$ but the second, third nearest reference vector $\boldsymbol{u}_{i_1}$, $\boldsymbol{u}_{i_2}$, $\cdots$ are also updated in learning.

Let $D^* = \{\boldsymbol{x}^p | p \in Z_P\}$ for the set D. Let $p(\boldsymbol{v})$ be the probability distribution for $\boldsymbol{v} \in V$. Then, NG is introduced as follows [10]:

Learning Algorithm B* (Neural Gas)
Input : The set V of data
Output : The set U of reference vectors
Step B*1 : The initial values of reference vectors are set randomly. The learning coefficients ε_{int} and ε_{fin} are set, respectively. Let T_{max} and θ be the maximum number of learning time and the threshold, respectively.
Step B*2 : Let $t = 1$.
Step B*3 : Give a data $\boldsymbol{v} \in V$ based on $p(\boldsymbol{x})$ and neighborhood-ranking $k_i(\boldsymbol{v}, \boldsymbol{u}_i)$ is determined.
Step B*4 : Each reference vector $\boldsymbol{u}_i$ for $i \in Z_r$ is updated based on Eq.(13)
Step B*5 : If $t \geq T_{max}$, then the algorithm terminates and the set $U = \{\boldsymbol{u}_i | i \in Z_r\}$ of reference vectors is obtained. Otherwise go to Step B*3 as $t \leftarrow t + 1$.

Likewise, k-means method is introduced.

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.

By using Learning Algorithm B*, the learning method of the fuzzy system is introduced as follows [8]. In this case, assume that the distribution of the set D^* of learning data is a discrete uniform one. Let n_0 be the initial number of rules and $n = n_0$.

Learning Algorithm B (Fuzzy modeling of using NG)
Input : Learning data $\boldsymbol{D}^* = \{(x_1^p, \cdots, x_m^p) | p \in Z_P\}$
Output : Parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$
Step B1 : For learning data D^*, Learning Algorithm B* is performed and the set U of reference vectors is obtained, where $|U| = n$.
Step B2 : Each element of the set U is set to the center parameter $\boldsymbol{c}$ of each fuzzy rule. Further, the width parameter b_{ij} is set as follows :

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

where C_i and m_i are the set of learning data belonging to the i-th cluster and its cardinal number, respectively. Each initial value of w_i is selected randomly. Let $t = 1$.
Step B3 : The Steps A3 to A8 of learning algorithm A are performed.
Step B4 : If $t \geq T_{max}$ and $E(t) > \theta$, then go to Step B1 with $n \leftarrow n + 1$.

Likewise, fuzzy modeling using k-means is proposed.

C. Adaptive local linear mapping

In this section, the learning method to adaptively approximate the function $y = f(\boldsymbol{v})$ with $\boldsymbol{v} \in V \subseteq R^d$ and $y \in R$ using NG is introduced based on Ref. [10]. The set V denotes the function's domain. Let n be the number of 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.(11), and the coefficients a_{i0} and $\boldsymbol{a}_i$ define a linear mapping

$$g(\boldsymbol{v}) = a_{i0} + \boldsymbol{a}_i(\boldsymbol{v} - \boldsymbol{u}_i) \qquad (17)$$

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

$$g(\boldsymbol{v}) = a_{i(\boldsymbol{v})0} + \boldsymbol{a}_{i(\boldsymbol{v})}(\boldsymbol{v} - \boldsymbol{u}_{i(\boldsymbol{v})}) \qquad (18)$$

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 learning steps by approximating $D = \{(\boldsymbol{x}^p, y_p^r) | p \in Z_P\}$ is performed. In order to obtain the output coefficients a_{i0} and $\boldsymbol{a}_i$, the mean squared error $\frac{1}{|V|}\sum_{\boldsymbol{v} \in V}(f(\boldsymbol{v}) - g(\boldsymbol{v}))^2$ between the desirable and the obtained output, averaged over subregion $\boldsymbol{V}_i$, to be minimal is required for each i. Gradient descent with respect to a_{i0} and $\boldsymbol{a}_i$ yields [10].

$$\begin{aligned} \triangle a_{i0} &= \varepsilon' h_{\lambda'}(k_i(\boldsymbol{v}, \boldsymbol{u}_i))(y - a_{i0} - \boldsymbol{a}_i(\boldsymbol{v} - \boldsymbol{u}_i)) \qquad (19) \\ \triangle \boldsymbol{a}_i &= \varepsilon' h_{\lambda'}(k_i(\boldsymbol{v}, \boldsymbol{u}_i))(y - a_{i0} - \boldsymbol{a}_i(\boldsymbol{v} - \boldsymbol{u}_i))(\boldsymbol{v} - \boldsymbol{u}_i) \\ & \qquad (20) \end{aligned}$$

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

In order to apply the method with adaptively local linear mapping (LLM) to fuzzy modeling of the simplified fuzzy system, all parameters $\boldsymbol{a}_i$'s are not used. That is, the constants a_{i0}'s only are updated.

The algorithm is introduced as follows :

Learning Algorithm C
Input : Learning data $D = \{(\boldsymbol{x}^p, y_p) | p \in Z_P\}$ and $D^* = \{\boldsymbol{x}^p | p \in Z_P\}$.
Output : The set U of reference vectors and the coefficient a_{i0} for $i \in Z_n$.
Step C1 : The set U of reference vectors is determined using D^* by Algorithm B*. The subregions V_i for $i \in Z_n$ is determined using U, where V_i is defined by Eq.(11), $V^d = \cup_{i=1}^n V_i$ and $V_i \cap V_j = \emptyset$ $(i \neq j)$.
Step C2 : Each parameter $a_{i0} (i \in Z_n)$ is 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 : Each parameter a_{i0} for $i \in Z_n$ is updated based on the following Eq.(21).

$$a_{i0}(t + 1) = a_{i0}(t) + \varepsilon' h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i))(y - a_{i0}) \qquad (21)$$

Step C5 : If $t \geq T_{max}$, then the algorithm terminates else go to Step C3 with $t \leftarrow t + 1$, where T_{max} means the maximum number of learning times.

Remark that Algorithm C is one of the learning methods using NG and SDM [10]. Likewise, the learning methods using other VQ such as k-means and SDM are also considered.

D. The appearance probability of learning data based on the rate of change of output

Learning Algorithm B is a method that determines the initial assignment of fuzzy rules by vector quantization using the set D^*. In the previous paper, we proposed a method using both input and output parts of $\boldsymbol{D}$ to determine the initial assignment of parameters of the antecedent part of fuzzy rule [8].

Based on Ref. [8], the appearance probability is defined as follows :

Algorithm for Appearance Probability (Algorithm AP)
Input : Learning data $\boldsymbol{D} = \{(x_1^p, \cdots, x_m^p, y_p^r)|p \in Z_P\}$ and $\boldsymbol{D}^* = \{(x_1^p, \cdots, x_m^p)|p{\in}Z_P\}$
Output : Appearance probability $p_M(\boldsymbol{x})$ for $\boldsymbol{x}{\in}\boldsymbol{D}^*$
Step 1 : Give an input data $\boldsymbol{x}_i{\in}\boldsymbol{D}^*$, 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 degree of change of inclination of the output around output data to input data $\boldsymbol{x}^i$, by the following equation:

$$H(\boldsymbol{x}^i) = \sum_{l=1}^{M} \frac{|y^i - y^{i_l}|}{||\boldsymbol{x}^i - \boldsymbol{x}^{i_l}||} \tag{22}$$

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

$$p_M(\boldsymbol{x}^i) = \frac{H(\boldsymbol{x}^i)}{\sum_{j=1}^{P} H(\boldsymbol{x}^j)} \tag{23}$$

The method is called Algorithm AP.

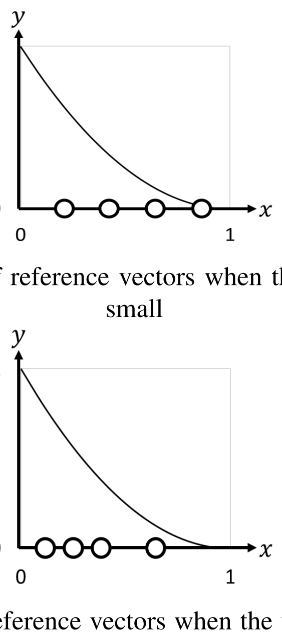
[Example 1]
Let us explain the computation about $p_M(\boldsymbol{x})$ in the case of $y = x^2 - 2x + 1$. The set of $D = \{(0.1{\times}k, 0.01{\times}k^2 - 0.2{\times}+1)|k \in Z_{10}^*\}$ and $D^* = \{0.1{\times}k|k \in Z_{10}^*\}$ are learning data. The gradient of $y = x^2 - 2x + 1$ is large around $x = 0$ and small around $x = 1$. If the gradient of the function is large in an area, it is desired to be set many rules. To achieve it, the probability $p_M(x)$ is desired to be large. If the probability $p_M(x)$ is large, the learning data $x{\in}D$ is chosen with high frequency. That is, if the gradient of the function is large around the learning data $x{\in}D$, the probability $p_M(x)$ is desired to be large to chose the learning data x with high frequency. Let us show an example when $M = 3$. $H(x_1)$ for the data $(x_1, y_1) = (0, 1)$ is calculated by the following calculation.

$$H(x_1) = \frac{|1-0.81|}{|0-0.1|} + \frac{|1-0.64|}{|0-0.2|} + \frac{|1-0.49|}{|0-0.3|} = 5.4$$

For the learning data $(x_2, y_2) = (0.1, 0.81)$, $(x_3, y_3) = (0.2, 0.64)$, $\cdots$, the values $H(x_2) = 5.2$, $H(x_3) = 5$, $\cdots$ are calculated by the same calculation. By using Eq.(23), $p_3(x_1) = 0.16$, $p_3(x_2) = 0.15$, $p_3(x_3) = 0.15$, $\cdots$ are obtained(shown in Table I). As shown in Table I, $H(x)$ is largest when $x = 0$ and smallest when $x = 1$. □

Learning algorithm F using Algorithm AP to fuzzy modeling is obtained as follows:

Learning Algorithm F
Input : Learning data $\boldsymbol{D} = \{(x_1^p, \cdots, x_m^p, y_p^r)|p \in Z_P\}$ and $\boldsymbol{D}^* = \{(x_1^p, \cdots, x_m^p)|p{\in}Z_P\}$
Output : Parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$.
Step F1 : The constants θ, T_{max}^0, T_{max} and M_0 for $1{\le}M_0$ are set. Let $M = M_0$. The probability $p_M(\boldsymbol{x})$ for $\boldsymbol{x}{\in}D^*$ is computed using Algorithm AP. The initial number n of rules is set.
Step F2 : The initial values of c_{ij}, b_{ij} and w_i are set randomly. Let $t = 1$.
Step F3 : Select a data $\boldsymbol{x}{\in}D^*$ based on $p_M(\boldsymbol{x})$.
Step F4 : Update c_{ij} by Eq.(14) to approximate the set D^* by the set of center parameters c_{ij}'s of fuzzy rule.
Step F5 : If $t < T_{max}^0$, go to Step F3 with $t{\leftarrow}t + 1$, otherwise go to Step F6 with $t{\leftarrow}1$.
Step F6 : Determine b_{ij} by Eq.(16).
Step F7 : Let $p = 1$.
Step F8 : Given data $(\boldsymbol{x}^p, y_p^r){\in}\boldsymbol{D}$.
Step F9 : Calculate μ_i and y^* by Eqs.(2) and (4).
Step F10 : Update parameters w_i, c_{ij} and b_{ij} by Eqs.(7), (8) and (9).
Step F11 : If $p < P$ then go to Step F8 with $p{\leftarrow}p + 1$.
Step F12 : If $E > \theta$ and $t < T_{max}$ then go to Step F8 with $t{\leftarrow}t + 1$, where E is computed as Eq.(5), and if $E < \theta$ then the algorithm terminate, otherwise go to Step F2 with $n{\leftarrow}n + 1$.



(a) Positions of reference vectors when the value M is small

(b) Positions of reference vectors when the value M is large

Fig. 1. Positions of reference vectors for the function $y = x^2 - 2x + 1$

[Example 2]
Fig. 1 shows about examples of reference vectors determined by using $p_M(x)$. The set of learning data is same as Example 1. Initial positions of reference vectors are set randomly. If the value M is small, most of output data y is not sufficiently considered and after learning, reference vectors are set with equal intervals as shown in Fig. 1(a). If the value M is large,some reference vectors are set in the area where the gradient of the function is large as shown Fig. 1(b). The value M must be chosen appropriately for each problem.

□

In Algorithm F, the SDM processes of F7 to F12 are performed after the initial values of parameters $\boldsymbol{c}$ and $\boldsymbol{b}$ of F4 to F6 are set.

As shown in Ref. [8], learning algorithm F realizes that many rules are needed at or near the places where output changes rapidly in learning data. The probability $p_M(\boldsymbol{x})$ is one method to perform it [8]. See the Ref. [8] about the detailed explanation of Algorithms AP and F for the simplified fuzzy modeling. Learning Algorithm F is the method based on NG. Likewise, Learning Algorithm F is also proposed based on k-means.

III. The Proposed fuzzy modeling using NG and k-means

It is known that the assignment of the initial parameters by VQ can improve the performance of fuzzy modeling based on SDM in terms of accuracy and the number of rules. For TS fuzzy inference model, which is superior to the simplified fuzzy inference model in terms of accuracy, it has been already proposed to assign the initial values of the weight parameters in the consequent part of fuzzy rules by using NG and its effectiveness has been demonstrated [11]. However, for simplified fuzzy inference model, a similar approach has not been proposed, and most of the conventional initialization methods using VQ take into account only the parameters of the antecedent parts of fuzzy rules. Therefore, for simplified fuzzy inference model, a new learning method using the Learning Algorithm C is proposed as follows:

Learning Algorithm G
Input : Learning data $D = \{(\boldsymbol{x}^p, y_p)|\ p{\in}Z_P\}$ and $D^* = \{\boldsymbol{x}^p|p{\in}Z_P\}$.
Output : Parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$.
Step G1 : Let θ, T_{max1}, T_{max2} and T_{max3} be the threshold of inference error, the maximum numbers of learning time for NG and the maximum number of learning time for SDM, respectively.
Step G2 : From Algorithm AP, the probability $p_M(\boldsymbol{x})$ is computed using the set D of learning data.
Step G3 : From Algorithm B*, the set U of reference vectors is computed using $p_M(\boldsymbol{x})$.
Step G4 : From Algorithm C, parameters a_{i0}'s of LLM are computed using the set U.
Step G5 : Each element of the set U is set to the center parameter $\boldsymbol{c}$ of each fuzzy rule and the width $\boldsymbol{b}$ of each fuzzy rule is computed from Eq.(16). Each parameter a_{i0} of LLM is set to the initial weight w_i of fuzzy rule. Let $t = 1$.
Step G6 : Steps A3 to A7 of Algorithm A are performed for $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$.
Step G7 : Inference error $E(t)$ of Eq.(5) is calculated. If $E(t){\le}\theta$ or $t = T_{max3}$ then go to Step G8. If $E(t) > \theta$ and $t < T_{max3}$, then go to Step G6 with $t{\leftarrow}t + 1$.
Step G8 : If $E(t){\le}\theta$, then the algorithm terminates. If $E(t) > \theta$, then go to Step G3 with $n{\leftarrow}n + 1$.

In Algorithm G, initial values of parameters $\boldsymbol{c}$ and $\boldsymbol{b}$ using G2 and G3, and parameters $\boldsymbol{w}$ using G4 are considered, respectively. Further, the SDM processes of steps G5 to G7 are performed.

Let us explain Algorithm G using an example.
[Example 3]

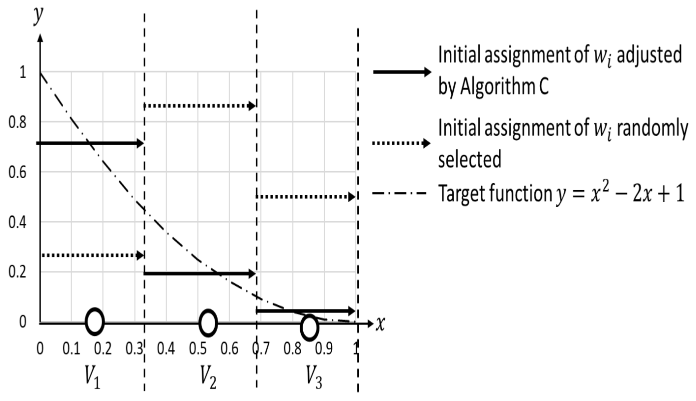


Fig. 2. The figure to explain an Example.

TABLE I
The appearance probability $p_3(\boldsymbol{x})$ for Example of $y = x^2 - 2x + 1$

x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	0.81	0.64	0.49	0.36	0.25	0.16	0.09	0.04	0.01	0
$p_3(x)$	0.16	0.15	0.15	0.13	0.11	0.09	0.08	0.06	0.04	0.02	0.02

The problem is to approximate the function $y = x^2-2x+1$ using Example 1.

From Steps G1 and G2, a probability $p_M(x)$ is computed as shown in Table I, where the number r of reference vectors is 3. From Step G3, NG is performed based on $p_M(x)$. Let $M = 3$. For example, two reference vectors are assigned in the interval from 0 to 0.7, and one reference vector is assigned for the remaining interval. Three regions V_1, V_2 and V_3 shown in Fig.2 are defined as Voronoi regions from the set U of reference vectors. From Step G4, a linear function is defined in each region. In the example, interval linear functions (constants) as shown in Fig.2 are obtained as the result of Algorithm C. If Algorithm C is not used, each linear function (constant) is given randomly as shown in Fig.2. From Step G5, the initial parameters of fuzzy rules are determined, respectively.

At Step G6, the conventional SDM is performed and parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are updated. If sufficient accuracy cannot be obtained, a fuzzy rule is adaptively added at Step G8. □

Learning Algorithm G is the method using NG as VQ. Likewise, Learning Algorithm is also proposed based on k-means.

IV. Numerical simulations

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

A. The conventional and proposed methods

In the following, the conventional algorithms are (a), (b), (c) and (d) and the proposed algorithm is (c'). The methods (a), (b), (c) and (c') are for simplified fuzzy inference model and (d) for TS fuzzy inference model [11]. Four the methods (b), (c) and (c'), learning algorithms based on k-means are also introduced. They are called (b-1), (c-1) and (c'-1), respectively.

(a) The method (a) is Learning Algorithm A, which is the primitive learning method for the simplified fuzzy system [1]. Initial parameters are set randomly and all parameters are updated using SDM until the inference error becomes sufficiently small.
(b) The method (b) is generalized one of learning method of RBF network [1]. The initial values of center parameters are determined using learning data D^* by NG and the width parameters are computed using the center parameters from Eq.16 [8]. The initial parameters of the weight are randomly selected. Further, all parameters are updated using SDM until the inference error becomes sufficiently small (See Fig.3(b)).
(c) The method (c) is Learning Algorithm F. Initial parameters of the center are determined using learning data D by NG and the initial parameters of the width are computed using the center parameters from Eq.16 [8]. The initial assignment of weight parameters is randomly selected. Further, all parameters are updated using SDM until the inference error becomes sufficiently small (See Fig.3(c)).
(c') The method (c') is Learning Algorithm G. In addition to the initial assignment of the method (c), the initial assignment of weight parameters of the consequent part is determined by Learning Algorithm C. Further, all parameters are updated using SDM until the inference error becomes sufficiently small (See Fig.3(c')).
(d) The method (d) is Algorithm E in Ref. [11] and is the version of TS fuzzy model of (c'). The simulation results are presented only for pattern classification problems.

Likewise, the methods (b-1), (c-1) and (c'-1) are proposed using k-means instead of NG.

B. Function approximation

The systems are identified by fuzzy inference systems. This simulation uses three systems specified by the following functions with 2 and 4-dimensional input space $[0,1]^2$ for Eq.(24) and $[-1,1]^4$ for Eqs.(25) and (26), respectively, and one output with the range $[0,1]$. Fig.4 shows the figure for Eq.(24). The numbers of Learning and Test data are 1000 and 1000 selected from the uniform random numbers, respectively.

$$y = \frac{\sin(10(x_1-0.5)^2+10(x_2-0.5)^2)+1}{2} \tag{24}$$

$$\begin{aligned} y &= \frac{(2x_1+4x_2^2+0.1)^2}{74.42} \\ &\quad + \frac{(3e^{3x_3}+2e^{-4x_4})^{-0.5}-0.077}{4.68} \end{aligned} \tag{25}$$

$$\begin{aligned} y &= \frac{(2x_1+4x_2^2+0.1)^2}{74.42} \\ &\quad \times \frac{(4\sin(\pi x_3)+2\cos(\pi x_4)+6)}{446.52} \end{aligned} \tag{26}$$

The constants θ, T_{max}, $K_{c_{ij}}$, $K_{b_{ij}}$ and K_{w_i} for each algorithm are 1.0×10^{-4}, 50000, 0.01, 0.01 and 0.1, respectively. The constants ε_{init}, ε_{fin} and λ for methods (b), (c) and (c') are 0.1, 0.01 and 0.7, respectively. The constants ε'_{init}, ε'_{fin} and λ' for method (c') are 0.1, 0.01 and 0.7, respectively. On methods (c) and (c'), the number M for the probability $p_M(\boldsymbol{x})$ is 500 for Eqs.(24) and (26) and 300 for Eq.(25), respectively. Fig.4 shows the figure for Eq.(24)

TABLE II
THE RESULT FOR FUNCTION APPROXIMATION FOR THE METHOD OF USING NG.

Method		Eq.(24)	Eq.(25)	Eq.(26)
(a)	the number of rules	5.00	11.60	6.20
	MSE for Learning($\times10^{-4}$)	0.34	0.07	0.03
	MSE of Test($\times10^{-4}$)	0.35	0.48	0.44
(b)	the number of rules	3.90	8.55	4.80
	MSE of Learning($\times10^{-4}$)	0.23	0.07	0.03
	MSE of Test($\times10^{-4}$)	0.27	0.09	0.03
(c)	the number of rules	3.75	7.75	4.65
	MSE of Learning($\times10^{-4}$)	0.22	0.07	0.03
	MSE of Test($\times10^{-4}$)	0.23	0.08	0.03
(c')	the number of rules	3.50	7.60	4.70
	MSE of Learning($\times10^{-4}$)	0.26	0.07	0.03
	MSE of Test($\times10^{-4}$)	0.27	0.09	0.03

In Table II, the number of rules and MSE's for learning and test are shown, where the number of rules means one when the threshold $\theta = 1.0\times10^{-4}$ of inference error is achieved in learning. The result of simulation is the average value from twenty trials. As a result, proposed method (c') reduces the number of rules compared to conventional methods while keeping the accuracy.

C. Classification problems

Iris, Wine, BCW and Sonar data from the UCI database are used for numerical simulation [13]. The numbers of data are 150, 178, 683 and 208, the numbers of input are 4, 13, 9 and 60 and the numbers of classes are 3, 3, 2 and 2, respectively. In this simulation, 5-fold cross-validation is used as the evaluation method. Threshold θ is 0.01 on Iris, 0.001 on Wine and 0.02 on BCW and Sonar, respectively. The constants T_{max}, $K_{c_{ij}}$, $K_{b_{ij}}$ and K_{w_i} for each method are 50000, 0.01, 0.01 and 0.1, respectively. The constants ε_{init}, ε_{fin} and λ for methods (b), (c), and (c') are 0.1, 0.01 and 0.7, respectively. The constants ε'_{init}, ε'_{fin} and λ' for methods (c') and (d) are 0.1, 0.01 and 0.7, respectively. On methods (c), (c') and (d), the number M for the probability $p_M(\boldsymbol{x})$ is 100 for Iris, Wine and Sonar and 200 for BCW, respectively.

Table III shows the result of classification for each method, where the number of rules means one when the threshold θ of inference error is achieved in learning. In Table III, the number of rules and RM's for learning and test are shown, where RM means the rate of misclassification. The result of simulation is the average value from twenty trials. It is shown that, among all the results, the proposed method (c') achieves the best or comparable performance in terms of both of accuracy of the number of rules. It should be noted that the proposed method (c') outperforms the conventional method (d) using TS fuzzy model for most cases.

D. Numerical simulations for the proposed method using k-means

Similar simulations are performed for the algorithm with k-means instead of NG. The parameters used for learning are the same except for λ, and the number of experiments is also the same. k-means is used for (b), (c), and (c') except (a). Table IV shows the results for the function approximation of

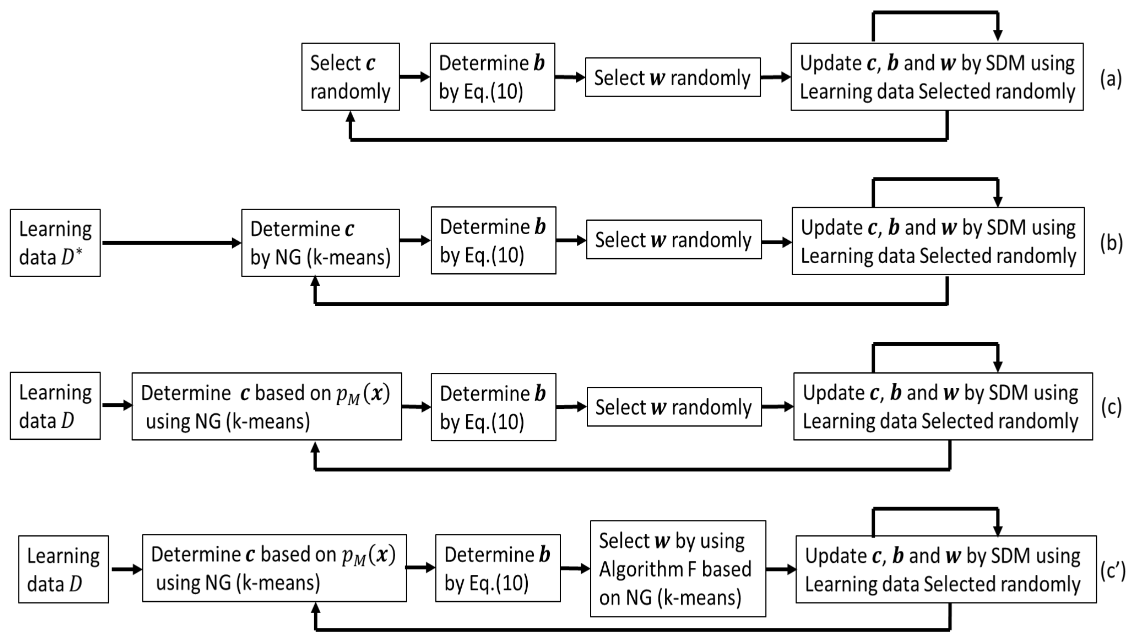

Select c randomly
Determine b by Eq.(10)
Select w randomly
Update c, b and w by SDM using Learning data Selected randomly
(a)
Learning data D^*
Determine c by NG (k-means)
Determine b by Eq.(10)
Select w randomly
Update c, b and w by SDM using Learning data Selected randomly
(b)
Learning data D
Determine c based on $p_M(x)$ using NG (k-means)
Determine b by Eq.(10)
Select w randomly
Update c, b and w by SDM using Learning data Selected randomly
(c)
Learning data D
Determine c based on $p_M(x)$ using NG (k-means)
Determine b by Eq.(10)
Select w by using Algorithm F based on NG (k-means)
Update c, b and w by SDM using Learning data Selected randomly
(c')

Fig. 3. Four charts of conventional and proposed algorithms : SDM and NG mean Steepest Descent Method and Neural Gas. NG (k-means) means to use NG or k-means. The feedback loop (the under arrow) means adding of a fuzzy rule, where $\boldsymbol{D} = \{(x_1^p, \cdots, x_m^p, y_p^r)|p \in Z_P\}$ and $\boldsymbol{D}^* = \{(x_1^p, \cdots, x_m^p)|p \in Z_P\}$.

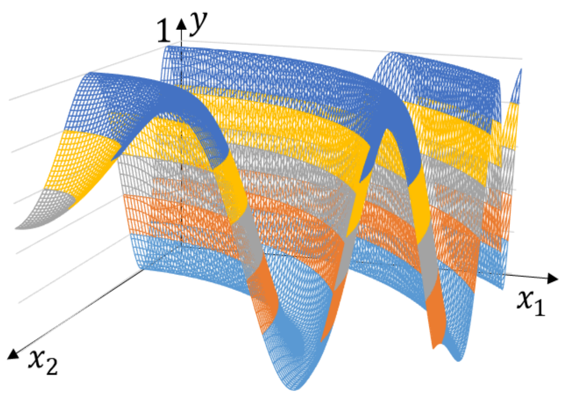


Fig. 4. The figure for the function $y = \frac{\sin(10(x_1-0.5)^2+10(x_2-0.5)^2)+1}{2}$ of Eq.(24).

TABLE III
THE RESULT FOR PATTERN CLASSIFICATION FOR THE METHOD OF USING NG.

Method		Iris	Wine	BCW	Sonar
(a)	the number of rules	2.34	4.14	2.60	7.11
	RM for Learning(%)	3.54	0.27	2.21	0.99
	RM of Test(%)	4.90	4.47	3.67	18.07
(b)	the number of rules	2.15	3.34	2.36	4.91
	RM of Learning(%)	3.42	0.30	2.23	0.67
	RM of Test(%)	4.40	3.33	3.66	17.26
(c)	the number of rules	2.10	3.26	2.05	2.61
	RM of Learning(%)	3.48	0.26	2.19	1.73
	RM of Test(%)	4.50	3.31	3.43	17.12
(c')	the number of rules	2.00	3.00	2.00	2.41
	RM of Learning(%)	3.50	0.35	2.19	1.81
	RM of Test(%)	4.60	3.14	3.41	16.20
(d)	the number of rules	2.0	2.0	2.0	2.8
	RM of Learning(%)	2.57	1.46	2.46	0.14
	RM of Test(%)	4.33	4.42	3.87	23.45

TABLE IV
THE RESULT FOR FUNCTION APPROXIMATION FOR THE METHOD OF USING K-MEANS.

Method		Eq.(24)	Eq.(25)	Eq.(26)
(a)	the number of rules	5.00	11.60	6.20
	MSE for Learning($\times 10^{-4}$)	0.34	0.07	0.03
	MSE of Test($\times 10^{-4}$)	0.35	0.48	0.44
(b-1)	the number of rules	4.30	8.45	6.00
	MSE of Learning($\times 10^{-4}$)	0.33	0.07	0.03
	MSE of Test($\times 10^{-4}$)	0.49	0.10	0.04
(c-1)	the number of rules	3.85	8.20	5.60
	MSE of Learning($\times 10^{-4}$)	0.28	0.07	0.04
	MSE of Test($\times 10^{-4}$)	0.29	0.09	0.05
(c'-1)	the number of rules	3.90	8.30	5.90
	MSE of Learning($\times 10^{-4}$)	0.22	0.08	0.03
	MSE of Test($\times 10^{-4}$)	0.25	0.10	0.04

TABLE V
THE RESULT FOR PATTERN CLASSIFICATION FOR THE METHOD OF USING K-MEANS.

Method		Iris	Wine	BCW	Sonar
(a)	the number of rules	2.34	4.14	2.60	7.11
	RM for Learning(%)	3.54	0.27	2.21	0.99
	RM of Test(%)	4.90	4.47	3.67	18.07
(b-1)	the number of rules	3.36	3.51	2.53	2.19
	RM of Learning(%)	4.1	0.3	1.7	2.1
	RM of Test(%)	5.1	3.0	16.5	3.6
(c-1)	the number of rules	3.05	3.09	2.61	2.05
	RM of Learning(%)	4.2	0.2	1.8	2.1
	RM of Test(%)	5.0	3.3	17.4	3.4
(c'-1)	the number of rules	3.07	3.12	2.51	2.04
	RM of Learning(%)	4.1	0.2	1.8	2.1
	RM of Test(%)	4.9	2.7	17.2	3.5

Eqs. (24), (25) and (26). Although (c) and (c') are superior to (a) and (b), there is no difference in ability between (c) and (c').

Similarly, Table V shows the results for the classification problem. In this case, the parameters used for learning are also the same except for λ, and the number of experiments is the same. In this case, (c) and (c') are superior to (a) and (b), but there is no difference in ability between (c) and (c').

Furthermore, let us consider the difference of performance when NG and k-means are used. Learning algorithm using NG is superior in the accuracy and the number of rules to the method using k-means, but the latter is superior in learning time to the former.

These should be selected according to the purpose.

V. Conclusion

In this paper, new learning methods of fuzzy modeling using NG and k-means with supervised learning (LLM) were proposed for simplified fuzzy inference model. Conventional learning methods using NG and k-means were to perform initial assignment only for antecedent part of fuzzy rules. In order to assign the initial values of weight parameters in the consequent part of fuzzy rules, the proposed method firstly divided the input space into Voronoi subregion by using NG (or k-means) and then assigned a constant to each subregion by using supervised learning (LLM). The simulation results on function approximation and pattern classification showed that the proposed method using NG achieved the best or comparable performance in terms of both of accuracy and the number of rules. Notably, the simulation results of pattern classification showed that the proposed method using NG for simplified fuzzy model was more effective than the similar approach for TS fuzzy model. From the result, it is considered that simpler model is suitable for this approach. Further, the proposed method was also superior to the method using generalized inverse matrix to determine the initial assignment of weight parameters [14]. Furthermore, the proposed method using NG was superior in accuracy and the number of rules to one using k-means. In future work, further improvement of fuzzy modeling using VQ will be considered and applied to other problems.

References

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

[2] 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.

[3] 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.

[4] 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, 2011, pp.894-913.

[5] H. Miyajima, N. Shigei and H. Miyajima, ”Fuzzy Inference Systems Composed of Double-Input Rule Modules for Obstacle Avoidance Problems,” IAENG International Journal of Computer Science, Vol. 41, Issue 4, 2014, pp.222-230.

[6] 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, 1998, pp.827-832.

[7] 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, 2005, pp.335-342.

[8] H. Miyajima, N. Shigei and H. Miyajima, ”Fuzzy Modeling using Vector Quantization based on Input and Output Learning Data,” Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2017, 15-17 March, 2017, Hong Kong, pp.1-6.

[9] W. Pedrycz, H. Izakian, ”Cluster-Centric Fuzzy Modeling,” IEEE Trans. on Fuzzy Systems, Vol. 22, Issue 6, 2014, pp. 1585-1597.

[10] 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, 1993, pp.558-569.

[11] H. Miyajima, N. Shigei and H. Miyajima, ”Fuzzy Modeling using Vector Quantization with Supervised Learning,” Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2018, 14-16 March, 2018, Hong Kong, pp.17-22.

[12] H. Miyajima, N. Shigei and H. Miyajima, ”Fuzzy Modeling using Neural Gas, World Congress on Engineering and Computer Science 2019,” Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2019, 22-24 October, 2019, San Francisco, USA, pp409-414.

[13] UCI Repository of Machine Learning Databases:,https://archive.ics.uci.edu/ml/index.php (Mar.20.2019).

[14] H. Miyajima, N. Shigei and H. Miyajima, ”Learning Algorithms for Fuzzy Inference Systems using Vector Quantization,” Intech Open, DOI: 10.5772/intechopen.79925, 2018.