Fuzzy Modeling using Vector Quantization with Supervised Learning

Hirofumi Miyajima, Noritaka Shigei, and Hiromi Miyajima

***Abstract*—It is known that learning methods of fuzzy modeling using vector quantization (VQ) and steepest descend method (SDM) are superior in the number of rules to other methods using only SDM. There are many studies on how to realize high accuracy with a few rules. Many methods of learning all parameters using SDM are proposed after determining initial assignments of the antecedent part of fuzzy rules by VQ using only input information, and both input and output information of learning data. Further, in addition to these initial assignments, the method using initial assignment of weight parameters of the consequent part of fuzzy rules is also proposed. Most of them are learning methods with simplified fuzzy inference, and little has been discussed with TS (Takagi Sugeno) fuzzy inference model. On the other hand, VQ method with supervised learning that divides the input space into Voronoi diagram by VQ and approximates each partial region with a linear function is known. It is desired to apply the method to TS fuzzy modeling using VQ. In this paper, we propose new learning methods of TS fuzzy inference model using VQ. Especially, learning methods using VQ with the supervised learning are proposed. Numerical simulations for function approximation, classification and prediction problems are performed to show the performance of proposed methods.**

***Index Terms*—Fuzzy Inference Systems, Vector Quantization, Neural Gas, Vector Quantization with Supervised Learning, Appearance Frequency.**

I. Introduction

With increasing interest in Artificial Intelligence (AI), many studies have been done with Machine Learning (ML). With ML, the supervised method such as BP learning for Multi-Layer Perceptron (MLP), the unsupervised one such as K-means method, and the Reinforcement Learning (RL) are well known. The learning method (fuzzy modeling) of fuzzy inference model is one of supervised ones, and many studies are conducted [1], [2]. 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 are problems of difficulty dealing with high dimensional spaces. 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 sufficiently large number of rules [3], 2) use GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) to determine fuzzy systems [4], 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 [5], and 4) use self-organization map (SOM) or VQ to determine the initial assignment of learning parameters [6], [7]. Especially, in learning methods of fuzzy inference model using VQ, there are many studies on how to realize high accuracy with a few rules [8], [9]. Many methods learning all parameters by SDM are proposed after determining the initial assignment of parameters of the antecedent part of fuzzy rules by VQ using only input information, and both input and output information of learning data. Further, in addition to these initial assignments, some methods using initial assignment of weight parameters of the consequent part of fuzzy rules are also proposed [8], [9]. Almost all of these are learning methods for simplified fuzzy inference model, and little has been discussed with TS fuzzy inference model [10]. On the other hand, a VQ method with the supervised learning that divides the input space into Voronoi diagram by VQ and approximates each partial region with a linear function is known [11]. It is desired to apply the method to determine the initial assignment of the parameters of TS fuzzy inference model to reduce the number of fuzzy rules.

In this paper, we propose new learning methods of TS fuzzy inference model using VQ. Especially, learning methods using VQ with the supervised learning are proposed. In Section II, conventional learning methods of TS fuzzy model and VQ are introduced. Further, VQ with the supervised learning is introduced. In Section III, new learning methods for TS fuzzy inference model using VQ are proposed. In Section IV, numerical simulations for function approximation, classification and prediction problems are performed to show the performance of proposed methods.

Although the proposed learning methods employ NG (Neural Gas) as a method of VQ, they also employ other VQ methods as well. Further, similar discussions can be developed for learning models other than the fuzzy inference model.

II. Preliminaries

A. The conventional TS and simplified fuzzy inference models

The conventional fuzzy inference model using SDM is described [1], [10]. 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 TS fuzzy inference model is expressed as

$$R_i : \text{if } x_1 \text{ is } M_{i1} \text{ and} \cdots x_j \text{is } M_{ij} \cdots \text{ and } x_m \text{ is } M_{im} \text{ then } y \text{ is } w_{i0} + w_{i1}x_1 + \cdots + w_{il}x_l + \cdots + w_{im}x_m, \quad (1)$$

where $i \in Z_n$ is a rule number, j and $l \in Z_m$ are variable numbers, M_{ij} is a membership function of the antecedent

Hirofumi Miyajima is with the Okayama University of Science, 1-1 Ridaicho, Kitaku, Okayama, 700-0005, 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.
This work was supported in part by the JSPS KAKENHI GRANT NUMBER JP17K00170.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

part, and w_{il} is the weight of the consequent part. The model is called TS (Takagi Sugeno) fuzzy inference model [10]. If $w_{il} = 0$ for $l \in Z_m$, then the model is called the simplified fuzzy inference model [1].

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

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

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

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

$$y^* = \frac{\sum_{i=1}^{n} \mu_i \left(\sum_{l=0}^{m} w_{il} x_l\right)}{\sum_{i=1}^{n} \mu_i} \tag{4}$$

where $x_0 = 1$.

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 [1].

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

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

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

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

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

$$\frac{\partial E}{\partial w_{il}} = \frac{\mu_i}{\sum_{i=1}^{n}\mu_i}(y^* - y^r)x_l \tag{7}$$

$$\frac{\partial E}{\partial c_{ij}} = \frac{\mu_i}{\sum_{i=1}^{n}\mu_i}(y^* - y^r)\left(\sum_{l=0}^{m} w_{il}x_l - y^*\right)\frac{x_j - c_{ij}}{b_{ij}^2} \tag{8}$$

$$\frac{\partial E}{\partial b_{ij}} = \frac{\mu_i}{\sum_{i=1}^{n}\mu_i}(y^* - y^r)\left(\sum_{l=0}^{m} w_{il}x_l - y^*\right)\frac{(x_j - c_{ij})^2}{b_{ij}^3} \tag{9}$$

where $x_0 = 1$, $i \in Z_n$, $j \in Z_m$ and $l \in Z_m^*$.

The conventional learning algorithm for TS model 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 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

Step A1 : The number of rules n is set to n_0. Let $t = 1$.
Step A2 : The parameters c_{ij}, b_{ij} and w_{il} 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 (4), μ_i and y^* are computed.
Step A6 : Parameters c_{ij}, b_{ij} and w_{il} are updated by Eqs.(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 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 methods

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:

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

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

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

The evaluation function for the partition is defined as follows:

$$E = \sum_{\boldsymbol{u}_i \in U}\sum_{\boldsymbol{v} \in V} \frac{h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i))}{\sum_{\boldsymbol{u}_l \in U} h_\lambda(k_l(\boldsymbol{v}, \boldsymbol{u}_l))} ||\boldsymbol{v} - \boldsymbol{u}_{i(\boldsymbol{v})}||^2 \tag{12}$$

For neural gas method [11], 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

$$||\boldsymbol{v} - \boldsymbol{u}_j|| < ||\boldsymbol{v} - \boldsymbol{u}_{i_k}|| \tag{13}$$

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

$$\triangle \boldsymbol{u}_i = \varepsilon h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i))(\boldsymbol{v} - \boldsymbol{u}_i) \tag{14}$$

$$h_\lambda(k_i(\boldsymbol{v}, \boldsymbol{u}_i)) = \exp\left(-k_i(\boldsymbol{v}, \boldsymbol{u}_i)/\lambda\right) \tag{15}$$

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

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 [11]. Otherwise, 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 $p(\boldsymbol{v})$ be the probability distribution of data vectors for V. Then, NG method is introduced as follows [11]:

Learning Algorithm B* (Neural Gas Method)

Step B*1 : The initial values of reference vectors are set randomly. The learning coefficients ε_{int} and ε_{fin} are set. Let T_{max} and θ be the maximum number of learning time and the threshold, respectively.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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.(14)
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 for $\boldsymbol{V}$ is obtained. Otherwise go to Step B*3 as $t\leftarrow t+1$.

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 fuzzy systems is introduced as follows [8] : 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 and $n = n_0$.

Learning Algorithm B (Learning of fuzzy inference system using NG)
Step B1 : For learning data D^*, Learning Algorithm B* is performed, where $|U| = n$.
Step B2 : Each element of the set U is set to the center parameter $\boldsymbol{c}$ of each fuzzy rule.
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 set of learning data belonging to the i-th cluster and its number, respectively. Each initial value of w_{il} 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 $n\leftarrow n+1$.

C. Adaptively local linear mapping

The aim in this section is to adaptively approximate the function $y = f(\boldsymbol{v})$ with $\boldsymbol{v}\in V\subseteq R^d$ and $y\in R$ using VQ [11]. The set V denotes the function's domain. Let us consider n computational units, each containing a reference vector $\boldsymbol{u}_i$ together with a constant a_{i0} and d-dimensional vectors $\boldsymbol{a}_i$. Learning Algorithm B* assigns each unit i to a subregion V_i as defined in Eq.(11), and the coefficients a_{i0} and $\boldsymbol{a}_i$ define a linear mapping

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

from R^d to R over each of the Voronoi diagram V_i (See Fig.1). Hence, the function $y = f(\boldsymbol{v})$ is approximated by $\tilde{y} = g(\boldsymbol{v})$ with

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

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 training steps by presenting $D = \{(\boldsymbol{x}^p, y_p^r) | p\in Z_P\}$ is performed. In order to obtain the output coefficients a_{i0} and $\boldsymbol{a}_i$, the mean squared error $\sum_{\boldsymbol{v}\in V} \left(f(\boldsymbol{v}) - g(\boldsymbol{v})\right)^2$ between the actual and the obtained output, averaged over subregion $\boldsymbol{V}_i$, to be minimal is required for each i. A gradient descent with respect to a_{i0} and $\boldsymbol{a}_i$ yields [11].

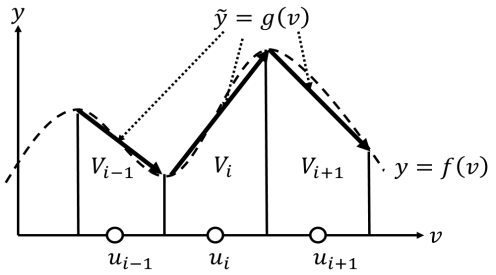

Fig. 1. The concept of local linear mapping : The set V_i is one composed of element $\boldsymbol{v}$ closest to the reference vector u_i. The interval V_i is approximated by a local linear mapping.

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

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

The method means VQ with the supervised learning.

The algorithm is introduced as follows [11] :

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 coefficients a_{i0} and $\boldsymbol{a}_i$ 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), $V^d = \cup_{i=1}^r V_i$ and $V_i \cap V_j = \emptyset$ $(i\neq j)$.
Step C2 : Parameters a_{i0} and $\boldsymbol{a}_i$ are set randomly. Let $t = 1$.
Step C3 : A learning data $(\boldsymbol{x}, y)\in D$ is selected based on $p(\boldsymbol{x})$. The rank $k_i(\boldsymbol{x}, \boldsymbol{u}_i)$ of $\boldsymbol{x}$ for the set V_i is determined.
Step C4 : Parameters a_{i0} and $\boldsymbol{a}_i$ for $i\in Z_r$ are updated based on Eqs.(19) and (20).
Step C5 : If $t\geq T_{max}$, then the algorithm terminates else go to Step C3 with $t\leftarrow t+1$.

Remark that Algorithm C is one of learning methods using NG and SDM [11].

D. The appearance frequency 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 this case, the set of output in learning data $\boldsymbol{D}$ is not used to determine the initial assignment of fuzzy rules. In the previous paper, we proposed a method using both input and output data to determine the initial assignment of parameters of the antecedent part of fuzzy rules [8].

Based on Ref. [8], the appearance frequency is defined as follows : Let $\boldsymbol{D}$ and $\boldsymbol{D}^*$ be the sets of learning data defined in Section 2.1.

Algorithm for Appearance Frequency (Algorithm AF)
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

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of inclination of the output around output data to input data $\boldsymbol{x}^i$, by the following equation:

$$H(\boldsymbol{x}^i) = \sum_{l=1}^{M} \frac{|y^i - y^{i_l}|}{||\boldsymbol{x}^i - \boldsymbol{x}^{i_l}||} \tag{21}$$

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 frequency (probability) $p_M(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i$ by normalizing $H(\boldsymbol{x}^i)$.

$$p_M(\boldsymbol{x}^i) = \frac{H(\boldsymbol{x}^i)}{\sum_{j=1}^{P} H(\boldsymbol{x}^j)} \tag{22}$$

The method is called Algorithm AF.

Learning algorithm D using Algorithm AF to TS fuzzy modeling is obtained as follows:

Learning Algorithm D

Step D1 : The constants θ, T_{max}^0, T_{max} and M_0 for $1 \leq M_0$ are set. Let $M = M_0$. The probability $p_M(\boldsymbol{x})$ for $\boldsymbol{x} \in D^*$ is computed using Algorithm AF. The initial number n of rules is set.

Step D2 : The initial values of c_{ij}, b_{ij} and w_{il} are set randomly.

Step D3 : Select a data $(\boldsymbol{x}^p, y_p)$ based on $p_M(\boldsymbol{x})$.

Step D4 : Update c_{ij} by Eq.(14).

Step D5 : If $t < T_{max}^0$, go to Step D3 with $t \leftarrow t+1$, otherwise go to Step D6 with $t \leftarrow 1$.

Step D6 : Determine b_{ij} by Eq.(16).

Step D7 : Let $p = 1$.

Step D8 : Given a data $(\boldsymbol{x}^p, y_p^r) \in \boldsymbol{D}$.

Step D9 : Calculate μ_i and y^* by Eqs.(2) and (4).

Step D10 : Update parameters w_{il}, c_{ij} and b_{ij} by Eqs.(7), (8) and (9).

Step D11 : If $p < P$ then go to Step D8 with $p \leftarrow p+1$.

Step D12 : If $E > \theta$ and $t < T_{max}$ then go to Step D8 with $t \leftarrow t+1$, where E is computed as Eq.(5), and if $E < \theta$ then the algorithm terminate, otherwise go to Step D2 with $n \leftarrow n+1$.

As shown in Ref. [8], learning algorithm D realizes that many rules are needed at or near the places where output changes rapidly in learning data. The probability $p_M(x)$ is one method to perform it [8]. See the Ref. [8] about the detailed explanation of Algorithms AF and D for the simplified fuzzy modeling.

III. Proposed methods of TS fuzzy modeling using VQ

It is known that learning methods for the simplified fuzzy model using VQ and SDM are effective in accuracy and the number of rules compared to other methods using only SDM. Further, it is also known that TS fuzzy inference model is superior in accuracy to the simplified fuzzy inference model [10]. Then, how is the performance of TS fuzzy modeling using VQ? The conventional methods learning all parameters in SDL is easily applied to TS fuzzy modeling after determining initial assignments of the antecedent part of fuzzy rules by VQ using only input information, and both input and output information of learning data (See the methods B and C). However, learning methods with weight parameters of the consequent part are difficult to apply to TS Fuzzy inference model. Therefore, we propose new learning methods using the Learning Algorithm C as follows:

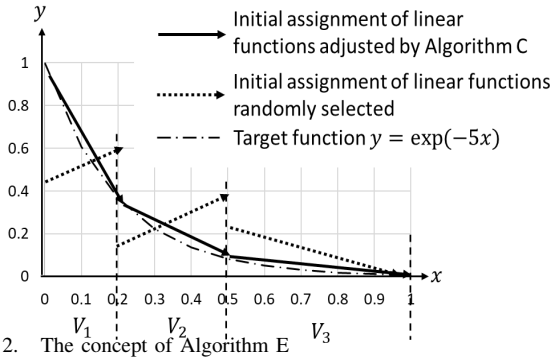

Fig. 2. The concept of Algorithm E

Learning Algorithm E

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 $\boldsymbol{w}$ of TS fuzzy inference model.

Step E1 : From Algorithm AF, the probability $p_M(\boldsymbol{x})$ is computed using D.

Step E2 : From Algorithm B*, the set U of reference vectors is computed using $p_M(\boldsymbol{x})$.

Step E3 : From Algorithm C, parameters a_{i0} and $\boldsymbol{a}_i$ of local linear mapping are computed using the set U.

Step E4 : 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.(19). Parameters a_{i0} and $\boldsymbol{a}_i$ of local linear mapping are set to the initial parameters of w_{i0} and $\boldsymbol{w}_i$ for fuzzy rules.

Step E5 : Let $t = 1$. Parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are updated from Step A3 to A8 of Algorithm A.

Step E6 : If $t > T_{max}$ and $E(t) > \theta_1$, then go to Step E2 with $n \leftarrow n+1$ as adding a fuzzy rule.

Let us explain Algorithm E using an example.

[Example]

The problem is that approximates the function $y = \exp(-5x)$ using ten learning data shown in Table I, where $D^* = \{0.1 \times k | k \in Z_{10}^*\}$ and $D = \{(x, \exp(-5x)) | x \in D^*\}$.

From Step E1, a probability $p_M(x)$ is formed as shown in Table I, where the number r of reference vectors is 3.

From Step E2, VQ is performed based on $p_M(x)$. Let $M = 3$. For example, two reference vectors are assigned in the interval from 0 to 0.4, 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 E3, a linear function is defined in each region. In the example, interval linear functions as shown in Fig.2 are obtained as the result of learning. If algorithm C is not used, each linear function is given randomly as shown in Fig.2.

From Steps E2 and E3, the initial parameters of the antecedent and the consequent parts of fuzzy rules are determined, respectively.

After Step E4, the conventional SDM is performed and parameters $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are updated. If sufficient accuracy cannot be obtained, fuzzy rules are adaptively added.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
LEARNING DATA FOR EXAMPLE OF $y = \exp(-5x)$

x	y	$p_M(x)$
0	1	0.272398
0.1	0.606531	0.23171
0.2	0.367879	0.196732
0.3	0.22313	0.119324
0.4	0.135335	0.072374
0.5	0.082085	0.043897
0.6	0.049787	0.026625
0.7	0.030197	0.016149
0.8	0.018316	0.009795
0.9	0.011109	0.005941
1	0.006738	0.005057

In order to compare the proposed methods with conventional ones, the following learning methods for TS model are used (See Fig.3):
(A) The method A is Learning Algorithm A, that is the conventional learning method for TS model. Initial parameters of $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ 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 networks [1]. Initial values of $\boldsymbol{c}$ are determined using the set $\boldsymbol{D}^*$ by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters $\boldsymbol{w}$ are randomly selected. Further, all parameters are updated using SDM until the inference error becomes sufficiently small.
(B') The method B' is the proposed one. In addition to the initial assignment of the method B, the initial assignment of weight parameters of the consequent part is determined by Algorithm C. Further, all parameters are updated using SDM until the inference error becomes sufficiently small.
(C) The method C is Learning Algorithm D. Initial values of $\boldsymbol{c}$ are determined using the set D by VQ and $\boldsymbol{b}$ is computed using $\boldsymbol{c}$. Weight parameters are randomly selected. Further, all parameters are updated using SDM until the inference error becomes sufficiently small.
(C') The method C' is the Algorithm E. In addition to initial assignment of the method C, the initial assignment of weight parameters of the consequent part is determined by the Algorithm C. Further, all parameters are updated using SDM until the inference error becomes sufficiently small.

IV. NUMERICAL SIMULATIONS

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

A. Function approximation

In order to show the effectiveness of proposed methods, numerical simulations of function approximation are performed. 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 $[0,1]^2$ (Eq.(23)) and $[-1,1]^4$ (Eqs.(24) and (25)), and one output with the range $[0,1]$. The number of Learning and Test data are 500 and 1000 for Eq.(23) and 512 and 6400

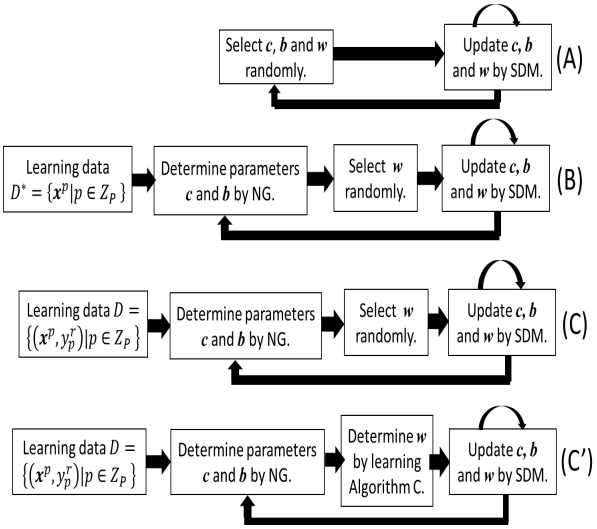

Fig. 3. Concept of conventional and proposed algorithms, where SDM andd NG mean Steepest Descent Method and Neural Gas method and the feedback loop means the adding of fuzzy rules.

TABLE II
THE RESULTS FOR FUNCTION APPROXIMATION

		Eq(23)	Eq(24)	Eq(25)
A	the number of rules	4.9	4.2	3.0
	MSE for Learning($\times 10^{-4}$)	0.37	0.31	0.20
	MSE of Test($\times 10^{-4}$)	0.49	0.46	0.22
B	the number of rules	4.1	4.0	3.0
	MSE of Learning($\times 10^{-4}$)	0.26	0.42	0.19
	MSE of Test($\times 10^{-4}$)	0.32	0.56	0.22
B'	the number of rules	4.1	4.1	3.1
	MSE of Learning($\times 10^{-4}$)	0.26	0.49	0.21
	MSE of Test($\times 10^{-4}$)	0.32	0.68	0.25
C	the number of rules	4.4	4.1	3.1
	MSE of Learning($\times 10^{-4}$)	0.29	0.49	0.21
	MSE of Test($\times 10^{-4}$)	0.47	0.68	0.25
C'	the number of rules	4.1	4.1	3.0
	MSE of Learning($\times 10^{-4}$)	0.26	0.54	0.18
	MSE of Test($\times 10^{-4}$)	0.35	0.76	0.21

for Eqs.(24) and (25), respectively.

$$y = \frac{\sin(10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2) + 1}{2} \quad (23)$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} + \frac{(3e^{3x_3} + 2e^{-4x_4})^{-0.5} - 0.077}{4.68} \quad (24)$$

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{74.42} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6)}{446.52} \quad (25)$$

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 Algorithms B, B', C and C' are 0.1, 0.01 and 0.7, respectively. The constants ε'_{init}, ε'_{fin} and λ' for Algorithms B' and C' are 0.1, 0.01 and 0.7, respectively.

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 methods B' and C' reduce the number of rules compared to conventional methods.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE III
THE DATASET FOR PATTERN CLASSIFICATION

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

TABLE IV
THE RESULT FOR PATTERN CLASSIFICATION

		Iris	Wine	BCW	Sonar
A	the number of rules	2.2	2.2	2.1	3.2
	RM for Learning(%)	2.8	1.6	2.4	0.4
	RM of Test(%)	3.5	6.1	3.8	19.4
B	the number of rules	2.1	2.1	2.4	5.6
	RM of Learning(%)	3.0	1.5	2.4	0.4
	RM of Test(%)	4.5	6.1	3.9	22.6
B'	the number of rules	2.2	2.1	2.2	5.4
	RM of Learning(%)	3.0	1.5	2.4	0.4
	RM of Test(%)	4.3	5.9	3.8	21.8
C	the number of rules	2.2	2.1	2.2	2.7
	RM of Learning(%)	3.0	1.5	2.4	0.2
	RM of Test(%)	4.3	5.9	3.8	24.3
C'	the number of rules	2.0	2.0	2.0	2.8
	RM of Learning(%)	2.6	1.5	2.5	0.1
	RM of Test(%)	4.3	4.4	3.9	23.5

B. Classification problems

Iris, Wine, BCW and Sonar data from UCI database shown in Table III are used for numerical simulation [12]. In this simulation, 5-fold cross-validation is used as the evaluation method. Threshold θ is 0.01 on Iris and Wine and 0.02 on BCW and Sonar, respectively. T_{max}, $K_{c_{ij}}$, $K_{b_{ij}}$ and K_{w_i} for each algorithm are 50000, 0.01, 0.01 and 0.1, respectively. ε_{init}, ε_{fin} and λ for Algorithms B, C, B' and C' are 0.1, 0.01 and 0.7, respectively. ε'_{init}, ε'_{fin} and λ' for Algorithms B' and C' are 0.1, 0.01 and 0.7, respectively.

Table IV shows the result of classification for each algorithm, where the number of rules means one when the threshold θ of inference error is achieved in learning. In Table IV, 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 the proposed method C' is superior in the number of rules to conventional methods.

C. Prediction of the Mackey-Glass Time Series

In this section, the prediction problem of time series generated by the Mackey-Glass equation is performed [11]. The prediction one requires to learn an input-output function $y = f(\boldsymbol{v})$ of a current state $\boldsymbol{v}$ of the time sequences into prediction of a future time series value y.

The Mackey-Glass equation is represented as follows :

$$\frac{\partial x(t)}{\partial t} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{26}$$

where $\alpha = 0.2$, $\beta = -0.1$ and $\tau = 17$. $x(t)$ is qusi-periodic and chaotic with a fractal attractor dimension 2.1 for the parameters.

In this simulation, let training pairs $\boldsymbol{v} = (x(t), x(t-6), x(t-12), x(t-18), y = x(t+6))$, that is, 4 inputs and one output. The numbers of learning and test data are 1000 and 1000, generated by Eq.(26), respectively. After learning using 1000 data, data of 1000 steps are predicted. The function is identified by fuzzy inference systems. θ, T_{max}, $K_{c_{ij}}$, $K_{b_{ij}}$ and K_{w_i} for each algorithm are 1.0×10^{-5}, 50000, 0.01, 0.01 and 0.1, respectively. ε_{init}, ε_{fin} and λ for Algorithms B, B', C and C' are 0.1, 0.01 and 0.7, respectively. ε'_{init}, ε'_{fin} and λ' for Algorithms B' and C' are 0.1, 0.01 and 0.7, respectively.

TABLE V
THE RESULT FOR FUNCTION APPROXIMATION

	A	B	B'	C	C'
the number of rules	9.6	6.1	5.8	5.0	5.1
MSE of Learning($\times10^{-5}$)	0.83	0.85	0.80	0.84	0.77
MSE of Test	0.103	0.103	0.102	0.103	0.103

The result of simulation is shown in Table V. The result of simulation is the average value from twenty trials. The result of Table V shows that proposed methods are in the number of rules superior to conventional ones.

V. CONCLUSION

In this paper, we proposed new learning methods of TS fuzzy inference model using VQ. Especially, learning methods using VQ with the supervised learning were proposed. With TS fuzzy modeling, conventional methods learning all parameters in SDL were proposed after determining initial assignments of the antecedent part of fuzzy rules by VQ using only input information, and both input and output information of learning data. In addition to these initial assignments, the methods determining weight parameters of linear functions of the consequent part of fuzzy rules were proposed. In numerical simulations for function approximation, classification and prediction problems, proposed methods were superior in the number of rules to conventional ones.

In the future work, other learning methods using VQ in TS fuzzy modeling will be proposed and other SDM methods using VQ with the supervised learning will be considered.

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, pp.687-694, 1995.
[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, pp.894-913, 2011.
[5] 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.
[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, pp.827-832, 1998.
[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, pp.335-342, 2005.
[8] H. Miyajima, N. Shigei and H. Miyajima, Fuzzy Modeling using Vector Quantization based on Input and Output Learning Data, International MultiConference of Engineers and Computer Scientists 2017, Vol.I, pp.1-6, Hong Kong, March, 2017.
[9] W. Pedrycz, H. Izakian, Cluster-Centric Fuzzy Modeling, IEEE Trans. on Fuzzy Systems, Vol. 22, Issue 6, pp. 1585-1597, 2014.
[10] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. on Systems, Man, and Cybernetics, Vol.SMC-15, No.1, pp.116-132, 1985.
[11] 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.
[12] UCI Repository of Machine Learning Databases and Domain Theories, ftp://ftp.ics.uci.edu/pub/machinelearning-Databases.

ISBN: 978-988-14047-8-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)