Proceedings of the International MultiConference of Engineers and Computer Scientists 2016 Vol I,
IMECS 2016, March 16 - 18, 2016, Hong Kong

Fast Learning Algorithm for Fuzzy Inference Systems using Vector Quantization

Hirofumi Miyajima^{†_1}, Noritaka Shigei^{†_2}, and Hiromi Miyajima^{†_3}

***Abstract*—It is known that learning methods of fuzzy inference systems using vector quantization (VQ) and steepest descend method (SDM) are superior in terms of the number of rules. However, they need a great deal of learning time. The cause could be that both of VQ and SDM perform only local searches. On the other hand, it has been shown that a learning method of radial basis function (RBF) networks using VQ and generalized inverse method (GIM) is much fast. In this paper, we propose a new learning method using VQ, GIM and SDM. The method iterates three stages in the outer loop of the algorithm. The first stage adjust the fuzzy rule arrangement by using VQ, the second one determines the weights of fuzzy rules by using GIM, and the third one updates both of the rule arrangement and the weights. In order to demonstrate the validity of the proposed method, numerical simulations for function approximation and pattern classification problems are performed. Specifically, it is shown that the proposed method reduces the learning time to about one-tenth compared to conventional methods in function approximation problem.**

***Index Terms*—Fuzzy Inference Systems, Vector Quantization, Neural Gas, Generalized Inverse Method, Appearance frequency.**

I. Introduction

Many studies on self-tuning fuzzy systems have been made [1]–[3]. Their aim is to construct automatically fuzzy systems from learning data. Although most of the methods are based on steepest descend method (SDM), the obvious drawbacks of the method are its large time complexity and getting stuck in a shallow local minimum. Further, there is difficulty for learning with high dimensional spaces [4]–[6]. 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 [7], 2) use GA (Genetic Algorithm) and PSO (Particle Swarm Optimization) to determine the structure of the fuzzy model [8], 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 [10], [11], and 4) use a self-organization or a vector quantization technique to determine the initial assignment [9], [12], [13]. Specifically, learning methods using VQ and SDM are superior in the number of rules, but they need a great deal of learning time [17]. The cause could be that both of VQ and SDM perform only local searches. On the other hand, it has been shown that a learning method of radial basis function (RBF) networks using VQ and generalized inverse method (GIM) is much fast [3].

In this paper, we propose a new learning method using VQ, GIM and SDM. The method consists of three stages iterated in the outer loop of the algorithm. Each of three stages utilize either VQ, GIM or SDM for tuning the fuzzy inference system. The first stage adjusts the fuzzy rule arrangement by using VQ, the second one adjusts the weights of fuzzy rules by using GIM, and the last one adjusts both of the rule arrangement and the weights. In order to demonstrate the validity of the proposed method, numerical simulations for function approximation and pattern classification problems are performed. Specifically, it is shown that the proposed method reduces the learning time to about one-tenth compared to conventional methods in function approximation problem.

II. Preliminaries

A. The conventional fuzzy inference model

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

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

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

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

Let c_{ij} and b_{ij} denote the center and the width values of M_{ij}, respectively. If Gaussian membership function is used, then M_{ij} is expressed as follow:

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

The output y^* of fuzzy inference is calculated by Eq.(4).

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

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^*.

Affiliation: Graduate School of Science and Engineering, Kagoshima University, 1-21-40 Korimoto, Kagoshima 890-0065, Japan
corresponding auther to provide email: miya@eee.kagoshima-u-ac.jp
$\dagger_1$ email: k3768085@kadai.jp
$\dagger_2$ email: shigei@eee.kagoshima-u-ac.jp
$\dagger_3$ email: miya@eee.kagoshima-u-ac.jp

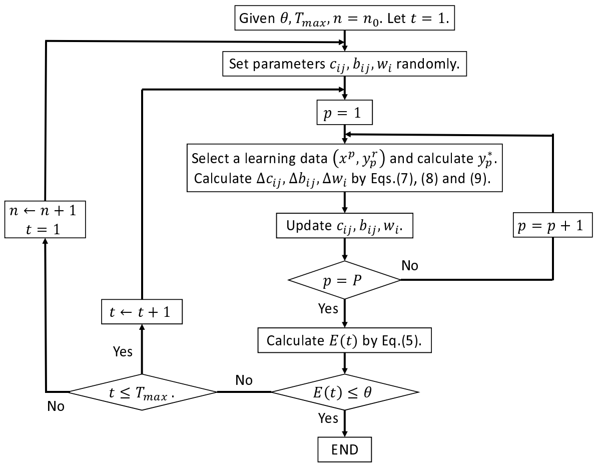
Given $\theta, T_{max}, n = n_0$. Let $t = 1$.
Set parameters c_{ij}, b_{ij}, w_i randomly.
$p = 1$
Select a learning data (x^p, y_p^r) and calculate y_p^*.
Calculate $\Delta c_{ij}, \Delta b_{ij}, \Delta w_i$ by Eqs.(7), (8) and (9).
Update c_{ij}, b_{ij}, w_i.
$p = P$
No
$p = p + 1$
Yes
Calculate $E(t)$ by Eq.(5).
$E(t) \le \theta$
Yes
END
No
$t \le T_{max}$.
Yes
$t \leftarrow t + 1$
No
$n \leftarrow n + 1$
$t = 1$

Fig. 1. The flowchart of the conventional learning algorithm

In this section, we describe the conventional learning algorithm [3].

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 $\alpha \in \{c_{ij}, b_{ij}, w_j\}$ is updated based on SDM as follows [1]–[3]:

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

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

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

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

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

The conventional learning algorithm is shown as Fig.1 [1]–[3], where θ and T_{max} are 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 $\boldsymbol{V} \subseteq \boldsymbol{R}^m$, utilizing only a finite set $\boldsymbol{C} = \{\boldsymbol{c}_i | i \in Z_r\}$ 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 $\boldsymbol{v} \in \boldsymbol{V}$ as follows:

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

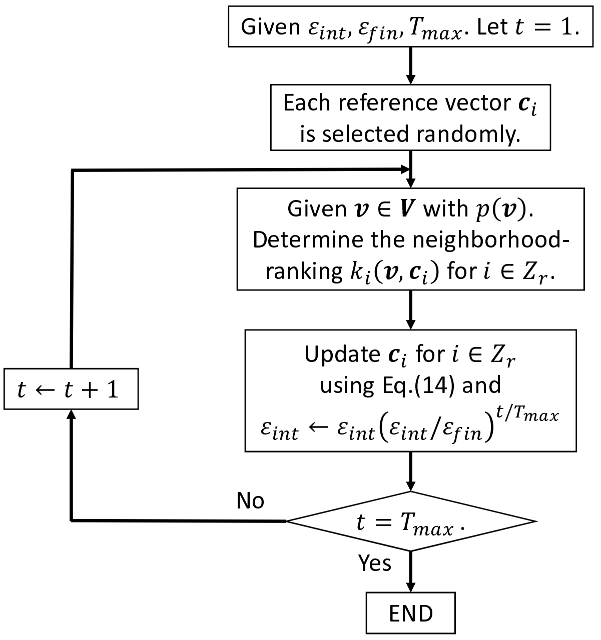
Given $\varepsilon_{int}, \varepsilon_{fin}, T_{max}$. Let $t = 1$.
Each reference vector $\boldsymbol{c}_i$ is selected randomly.
Given $\boldsymbol{v} \in \boldsymbol{V}$ with $p(\boldsymbol{v})$. Determine the neighborhood-ranking $k_i(\boldsymbol{v}, \boldsymbol{c}_i)$ for $i \in Z_r$.
Update $\boldsymbol{c}_i$ for $i \in Z_r$ using Eq.(14) and $\varepsilon_{int} \leftarrow \varepsilon_{int}(\varepsilon_{int}/\varepsilon_{fin})^{t/T_{max}}$
$t \leftarrow t + 1$
No
$t = T_{max}$.
Yes
END

Fig. 2. Neural Gas method

From the finite set $\boldsymbol{S}, \boldsymbol{V}$ is portioned as follows:

$$\boldsymbol{V}_i = \{\boldsymbol{v} \in \boldsymbol{V} | ||\boldsymbol{v} - \boldsymbol{c}_i|| \le ||\boldsymbol{v} - \boldsymbol{c}_j|| \ for \ j \in Z_r\} \tag{11}$$

The evaluation function for the partition is defined as follows:

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

For neural gas method [14], 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

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

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

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

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

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 [14]. 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 $\boldsymbol{V}$. The flowchart of the conventional neural gas algorithm is shown as Fig.2 [14], 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 [14].

By using Learning Algorithm NG, learning method of fuzzy systems is shown as follows [13] : 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 $\boldsymbol{D}^*$, Learning Algorithm NG is performed by using $\boldsymbol{D}^*$ as the set $\boldsymbol{V}$. As a result, the set $\boldsymbol{c}$ of inference vectors for $\boldsymbol{D}^*$ is made, where $|\boldsymbol{c}| = n_0$.

Step B2 : Each initial value c_{ij} is set to a reference vector. 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 set of element and the number of learning data belonging to the i-th cluster C_i. Each initial weight w_i is selected randomly. Further, Step A1 of Learning Algorithm A is performed.

Step B3 : The Steps A3 to A9 of learning algorithm A are performed.

C. Determination of weights using the generalized inverse method

Let us explain fuzzy inference systems and interpolation problem using the generalized inverse method [3]. This problem can be stated mathematically as follows:

Given P points $\{\boldsymbol{x}^p = (x_1^p, \cdots, x_m^p)\,|p\in Z_P\}$ and P real numbers $\{y_p^r|p\in Z_P\}$, find a function $f : \boldsymbol{R}^m \rightarrow \boldsymbol{R}$ such that the following conditions are satisfied :

$$f(\boldsymbol{x}^p) = y_p^r \; p\in Z_P \tag{17}$$

In the case of fuzzy inference system, this problem is solved as follows:

$$\begin{aligned} y_p = f(\boldsymbol{x}^p) &= \sum_{i=1}^{n} w_i \phi_{pi}(||\boldsymbol{x}^p - \boldsymbol{c}_i||) && (18) \\ \phi_{pi}(||\boldsymbol{x}^p - \boldsymbol{c}_i||) &= \frac{\mu_i}{\sum_{i=1}^{n} \mu_i} && (19) \end{aligned}$$

That is,

$$\Phi \boldsymbol{w} = \boldsymbol{y}, \tag{20}$$

where

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{P1} & \phi_{P2} & \cdots & \phi_{Pn} \end{bmatrix} \tag{21}$$

Let $P = n$ and $\boldsymbol{x}^i = \boldsymbol{c}_i$. The width parameters are determined by Eq.(19). Then, if $\phi_i(\cdot)$ is suitably selected as Gaussian function, then the solution of weights w is obtained as

$$\boldsymbol{w} = \Phi^{-1} \boldsymbol{y} \tag{22}$$

Let us consider the case $n < P$. This is the realistic case. The optimal solution $\boldsymbol{w}^*$ that minimizes $E = ||y^r - \Phi \boldsymbol{w}||^2$ can be obtained as follows :

$$\boldsymbol{w}^+ = \Phi^T \boldsymbol{y} \text{ and } E_{min} = ||(I - \Psi)\boldsymbol{y}||^2, \tag{23}$$

where $\Phi^+ \triangleq [\Phi^T\Phi]^{-1}\Phi^T$, $\Psi \triangleq \Phi\Phi^T$ and I is identify matrix of $P \times P$.

Φ^+ is called the generalized inverse of Φ. The method using Φ^+ to determine the weights is called the generalized inverse method (GIM) [3].

D. The appearance frequency of input 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 $\boldsymbol{D}^*$ of input for learning data. 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, Kishida proposed a method considering both input and output data to determine the initial assignment of fuzzy rules [12].

Based on the literature [12], the appearance frequency is defined as follows : Let $\boldsymbol{D}$ and $\boldsymbol{D}^*$ be the sets of learning data defined in 2.1.

Calculation Algorithm for the appearance frequency

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:

$$H(\boldsymbol{x}^i) = \sum_{l=1}^{M} \left| \frac{y^i - y^{i_l}}{||\boldsymbol{x}^i - \boldsymbol{x}^{i_l}||} \right|, \tag{24}$$

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

$$p_M(\boldsymbol{x}^i) = \frac{H(\boldsymbol{x}^i)}{\sum_{j=1}^{P} H(\boldsymbol{x}^j)} \tag{25}$$

Learning algorithm C using the appearance frequency is shown as follow [12]:

Learning Algorithm C

Step 1 : θ, T^0_{max}, T_{max}, n and M_0 for $1 \leq M_0$ are set. Initial value of c_{ij}, b_{ij} and w_i are set randomly. Let $M \leftarrow M_0$. The appearance frequency $p_M(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i \in \boldsymbol{D}^*$ is computed.

Step 2 : Select a data $(\boldsymbol{x}^p, y^p)$ based on $p_M(\boldsymbol{x}^p, y^p)$ for $1 \leq p \leq P$.

Step 3 : Update c_{ij} by Eq.(14).

Step 4 : If $t < T^0_{max}$, go to Step 2 with $t \leftarrow t+1$, otherwise go to Step 5 with $t \leftarrow 1$.

Step 5 : Determine b_{ij} by Eq.(16).

Step 6 : Let $p \leftarrow 1$.

Step 7 : Given a data $(\boldsymbol{x}^p, y_p^r) \in \boldsymbol{D}$.

Step 8 : Calculate μ_i and y^* by Eqs.(2) and (4).

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

Step 10 : If $p < P$ then go to Step 7 with $p \leftarrow p+1$.

Step 11 : If $E > \theta$ and $t < T_{max}$ then go to Step 7 with $t \leftarrow t+1$, where E is computed as Eq.(5), and if $E < \theta$ then the algorithm terminate, otherwise go to Step 7 with

$n \leftarrow n+1$ and initial value of c_{ij}, b_{ij} and w_i are set randomly, $M \leftarrow M_0$ and the appearance frequency $p_M(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i \in \boldsymbol{D}^*$ is computed.

III. The proposed method

It is shown that learning method C using VQ and SDM is effective in accuracy and the number of rules to other methods. However, it needs a great deal of learning time. The cause could be that both of VQ and SDM are local search methods. On the other hand, it has been shown that a learning method of RBF networks using VQ and GIM is much fast compared to other learning methods [3]. Specifically, the method using GIM seems to be effective compared to methods using SDM, because the method is not local search. However, the method using only VQ and GIM is not always effective [15]. Therefore, we propose a new learning method composed of three stages using VQ, GIM and SDM. The three stages are iterated in the outer loop of the algorithm. The first stage adjusts the center and width parameters by using VQ, the second one updates the weight parameters by using GIM, and the last one adjusts all three parameters by using SDM. With iterating processes, parameters of the result of SDM are set to ones of the next process if the inference error is improved.

The proposed method is shown as follows:

Learning Algorithm E

Step 1 : θ, T^0_{max}, T_{max}, M_0, M_{max} and β for $1 \leq M_{max}$ and $\beta < P$ are set. Let $M \leftarrow M_0$. Initial values of c^{best}_{ij}, b^{best}_{ij} and w^{best}_i are set randomly. Let E_{best} be the MSE for fuzzy inference system with c^{best}_{ij}, b^{best}_{ij} and w^{best}_i. Let $n \leftarrow n_0$.

(Stage 1: Learning by VQ)

Step 2 : Let $c_{ij} \leftarrow c^{best}_{ij}$, $b_{ij} \leftarrow b^{best}_{ij}$, $w_i \leftarrow w^{best}_i$ and $t = 1$.

Step 3 : Select a data $(\boldsymbol{x}^p, y^p)$ based on $p_M(\boldsymbol{x}^p, y^p)$ for $1 \leq p \leq P$.

Step 4 : Update c_{ij} by Eq.(14).

Step 5 : If $t < T^0_{max}$, 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).

(Stage 2: Learning by GIM)

Step 7 : Determine w_i by Eq.(23).

(Stage 3: Learning by SDM)

Step 8 : Let $p \leftarrow 1$.

Step 9 : Given a data $(\boldsymbol{x}^p, y^r_p) \in \boldsymbol{D}$.

Step 10 : Calculate μ_i and y^* by Eqs.(2) and (4).

Step 11 : Update parameters c_{ij}, b_{ij} and w_{ij} by Eqs.(7), (8) and (9).

Step 12 : If $p < P$ then go to Step 8 with $p \leftarrow p+1$.

Step 13 : If $E(t) > \theta$ and $t < T_{max}$ then go to Step 9 with $t \leftarrow t+1$, where E is computed as Eq.(5).

Step 14 : If $E(t) < E_{best}$ then $c^{best}_{ij} \leftarrow c_{ij}$, $b^{best}_{ij} \leftarrow b_{ij}$, $w^{best}_i \leftarrow w_i$ and $E_{best} \leftarrow E(t)$.

Step 15 : If $E(t) > \theta$ and $M < M_{max}$ then go to Step 2 with $M \leftarrow M+\beta$, else if $E(t) < \theta$, then the algorithm terminates, otherwise go to Step 2 with $n \leftarrow n+1$, $M \leftarrow M_0$ and c^{best}_{ij}, b^{best}_{ij} and w^{best}_i are set randomly.

In order to compare the proposed method with conventional ones, the following methods are used (See Fig.3):

(A) Method A is one based on the algorithm of Fig.1 [1], [2]. Initial parameters of $\boldsymbol{c}$, $\boldsymbol{b}$ and $\boldsymbol{w}$ are set randomly and all

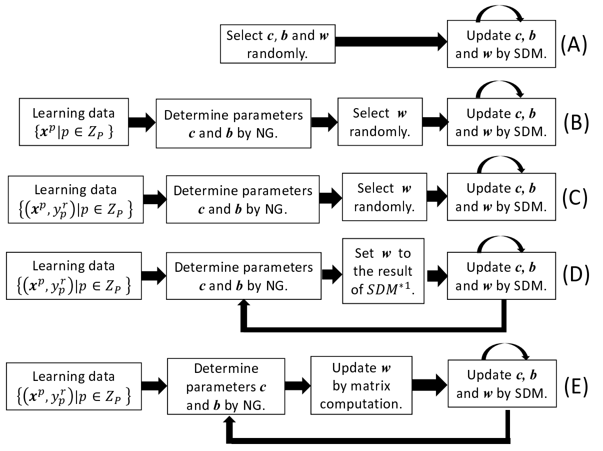
Select c, b and w randomly.
Update c, b and w by SDM.
(A)
Learning data {x^p | p ∈ Z_P }
Determine parameters c and b by NG.
Select w randomly.
Update c, b and w by SDM.
(B)
Learning data {(x^p, y^r_p) | p ∈ Z_P }
Determine parameters c and b by NG.
Select w randomly.
Update c, b and w by SDM.
(C)
Learning data {(x^p, y^r_p) | p ∈ Z_P }
Determine parameters c and b by NG.
Set w to the result of SDM*1.
Update c, b and w by SDM.
(D)
Learning data {(x^p, y^r_p) | p ∈ Z_P }
Determine parameters c and b by NG.
Update w by matrix computation.
Update c, b and w by SDM.
(E)

Fig. 3. Concept of conventional and proposed algorithms, where SDM and NG mean Steepest Descent Method and Neural Gas method, and the mark $*1$ means that initial values of $\boldsymbol{w}$ are selected randomly.

parameters are updated using SDM until the inference error become sufficiently small.

(B) Method B is known as learning method of RBF networks [3], [9], [13]. Initial values of $\boldsymbol{c}$ are determined using $\boldsymbol{D}^*$ by VQ and $\boldsymbol{b}$ is computed by Eq.(16). Weight parameters $\boldsymbol{w}$ are randomly selected. Further, all parameters are updated using SDM until the inference error become sufficiently small.

(C) Method C is proposed in Ref. [9]. Initial values of $\boldsymbol{c}$ are determined using $\boldsymbol{D}$ by VQ and $\boldsymbol{b}$ is computed by Eq.(16). Weight parameters are randomly selected. Further, all parameters are updated using SDM until the inference error become sufficiently small.

Note that the difference between Methods B and C is that learning data $\boldsymbol{D}$ or $\boldsymbol{D}^*$ is used in algorithm.

(D) Method D is proposed in Ref. [17]. It is learning method composed of iterating two stages. The center parameters $\boldsymbol{c}$ are determined using $\boldsymbol{D}$ by VQ and $\boldsymbol{b}$ is computed by Eq.(16). Weight parameters $\boldsymbol{w}$ is set to the results of SDM, where the initial values of $\boldsymbol{w}$ are set randomly. Further, all parameters are updated using SDM for the definite number of learning time. With iterating processes, parameters of the result of SDM are set to ones of the next process. Outer iterating process is repeated until the inference error become sufficiently small.

(E) Method E is the proposed one. It is learning method composed of iterating three stages. It starts by breaking the method into three stages: learning in the first stage, intermediate stage of adjusting the center and width parameters, and the next stage of updating the weight parameters using GIM. As for the final stage, three parameters are updated using SDM for the definite number of learning time. With iterating processes, parameters of the result of SDM are set to ones of the next process.

IV. Numerical simulations

In order to show the effectiveness of Learning Algorithm E, simulations of function approximation and classification problems are performed.

A. Performance of initial assignment of parameters

In this simulation, performance of initial assignment of parameters for Methods A, B and E*, where E* means

TABLE I
CONDITIONS OF ALGORITHMS FOR NUMERICAL SIMULATION OF FUNCTION APPROXIMATION

	A	B	C	D	E
$K_{c_{ij}}$	0.01	0.01	0.01	0.01	0.01
$K_{b_{ij}}$	0.01	0.01	0.01	0.01	0.01
K_{w_i}	0.1	0.1	0.1	0.1	0.1
ε_{init}		0.1	0.1	0.1	0.1
ε_{fin}		0.01	0.01	0.01	0.01
λ		0.7	0.7	0.7	0.7

TABLE II
THE RESULTS FOR FUNCTION APPROXIMATION

A	MSE for Learning($\times 10^{-4}$)	0.10
	MSE of Test($\times 10^{-4}$)	1.63
	t	438.1
B	MSE of Learning($\times 10^{-4}$)	0.10
	MSE of Test($\times 10^{-4}$)	1.86
	t	106.7
E	MSE of Learning($\times 10^{-4}$)	0.10
	MSE of Test($\times 10^{-4}$)	2.04
	t	90.3

TABLE III
CONDITIONS OF ALGORITHMS FOR NUMERICAL SIMULATION OF FUNCTION APPROXIMATION PROBLEMS

	A	B	C	D	E
T_{max}	50000	50000	50000	5000	50
$K_{c_{ij}}$	0.01	0.01	0.01	0.01	0.01
$K_{b_{ij}}$	0.01	0.01	0.01	0.01	0.01
K_{w_i}	0.1	0.1	0.1	0.1	0.1
ε_{init}		0.1	0.1	0.1	0.1
ε_{fin}		0.01	0.01	0.01	0.01
λ		0.7	0.7	0.7	0.7

TABLE IV
INITIAL PARAMETERS OF NUMERICAL SIMULATION FOR FUNCTION APPROXIMATION PROBLEMS OF EQS.(27), (28), (29) AND (30)

θ	1.0×10^{-4}
M	200
M_0	200
β	50
M_{max}	400
♯ Learning data	512
♯ Test data	6400

that the method E without outer iterating process and with iterating process of SDM until the inference error becomes sufficiently small.

The system is identified by fuzzy inference systems. This simulation uses four systems specified by the following functions with 4-dimensional input space $[0,1]^4$, and one output with the range $[0,1]$:

$$y = \frac{(2x_1 + 4x_2^2 + 0.1)^2}{37.21} \times \frac{(4\sin(\pi x_3) + 2\cos(\pi x_4) + 6)}{12} \tag{26}$$

The numbers M, M_0, β, M_{max} and the number of rules n are 200, 200, 0, 200 and 20, respectively. The threshold θ is 1.0×10^{-4}. Note that the number of rules is fixed. The results are average from ten trials. The results show that appropriate initial assignment of parameters results in a fast learning method.

B. Function approximation problems

In order to show the effectiveness of Learning Algorithm E, numerical simulations of function approximation are performed. 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.(27) and (28)) and $[-1,1]^4$((29) and (30)), and one output with the range $[0,1]$;

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

$$y = \frac{(\sin(2\pi x_1) \times \cos(x_2) \times \sin(\pi x_3) \times x_4 + 1.0)}{2.0} \tag{28}$$

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

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

Tables III and IV show the initial conditions for simulations. In Table III, A, B, C, D and E mean Learning Algorithms A, B, C, D and E. Table V shows the results for simulations. In Table V, the number of rules, MSE's for learning and test, and learning time(second) are shown, where the number of rules means one when the threshold $\theta = 1.0\times 10^{-4}$ of inference error is achieved in learning. The result of simulation is the average value from twenty trials. As a result, the proposed method E reduces the learning time to about one-tenth compared to other methods.

C. Classification problems

Iris, Wine and BCW data from UCI database shown in Table VI are used for numerical simulation [16]. In this

TABLE V
THE RESULTS FOR FUNCTION APPROXIMATION PROBLEMS

			Eq(27)	Eq(28)	Eq(29)	Eq(30)
A	The number of rules		4.2	13.6	7.2	5.1
	MSE	Learning	0.40	0.71	0.43	0.28
	($\times 10^{-4}$)	Test	0.52	1.09	1.00	0.49
	Learning time (s)		254.9	2617.5	777.9	365.7
B	The number of rules		5.6	14.9	5.2	3.7
	MSE	Learning	0.18	0.77	0.49	0.33
	($\times 10^{-4}$)	Test	0.27	1.42	1.11	0.48
	Learning time (s)		624.5	6766.1	513.0	255.5
C	The number of rules		4.8	15.6	5.5	4.0
	MSE	Learning	0.21	0.72	0.54	0.88
	($\times 10^{-4}$)	Test	0.34	1.33	0.69	0.53
	Learning time (s)		350.2	3125.0	447.6	210.0
D	The number of rules		3.0	8.4	4.0	3.0
	MSE	Learning	0.28	0.69	0.66	0.21
	($\times 10^{-4}$)	Test	0.35	1.25	0.76	0.23
	Learning time (s)		268.5	1673.3	425.9	267.0
E	The number of rules		3.0	8.0	8.6	5.5
	MSE	Learning	0.30	0.88	0.84	0.75
	($\times 10^{-4}$)	Test	0.39	1.25	1.15	1.02
	Learning time (s)		10.3	45.3	51.9	25.4

TABLE VI
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 VII
CONDITIONS OF ALGORITHMS FOR NUMERICAL SIMULATION OF PATTERN CLASSIFICATION

	A	B	C	D	E
T_{max}	50000	50000	50000	5000	50
$K_{c_{ij}}$	0.001	0.001	0.001	0.001	0.001
$K_{b_{ij}}$	0.001	0.001	0.001	0.001	0.001
K_{w_i}	0.05	0.05	0.05	0.05	0.05
ε_{init}		0.1	0.1	0.1	0.1
ε_{fin}		0.01	0.01	0.01	0.01
λ		0.7	0.7	0.7	0.7

TABLE VIII
INITIAL PARAMETERS FOR NUMERICAL SIMULATION OF PATTERN CLASSIFICATION

	Iris	Wine	BCW
θ	1.0×10^{-2}	1.0×10^{-2}	2.0×10^{-2}
M	100	100	200
M_0	30	40	200
β	10	10	30
M_{max}	120	130	470

TABLE IX
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
	learning time(s)	40.4	613.1	9161.4
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
	learning time(s)	16.8	7.4	9.6
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
	learning time(s)	38.9	529.0	2763.4
D	the number of rules	2.0	3.2	4.8
	RM of Learning(%)	3.3	1.5	1.6
	RM of Test(%)	4.0	6.7	3.8
	learning time(s)	25.1	204.8	1648.7
E	the number of rules	3.7	2.5	2.5
	RM of Learning(%)	3.3	1.1	1.3
	RM of Test(%)	3.8	6.5	2.1
	learning time(s)	8.6	3.3	37

simulation, 5-fold cross-validation is used. Tables VII and VIII show the initial conditions for simulations and Table IX shows the result of classification for each algorithm. In Table IX, the number of rules, RM's for learning and test, and learning time(second) are shown, where RM means the rate of misclassification. It is shown that the proposed method E is realized with high accuracy in a short time compared with other methods.

V. CONCLUSION

In this paper, we proposed a new learning method composed of iterating three stages. It started by breaking the method into three stages: learning in the first stage, intermediate stage adjusting the center and width parameters, and the next stage of updating the weight parameters using the generalized inverse method (GIM). As the final stage, three parameters were updated by learning based on SDM. In order to demonstrate the effectiveness of the proposed method, numerical simulations for function approximation and pattern classification problems were performed. It was shown that the proposed method reduces the learning time to about one-tenth compared to other methods in function approximation and is realized with high accuracy in a short time compared with other methods in classification problem.

In the future work, we will propose faster learning algorithm using VQ and the generalized inverse method compared to other methods.

REFERENCES

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

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

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

[4] J. Casillas, O. Cordon, F. Herrera and L. Magdalena, Accuracy Improvements in Linguistic Fuzzy Modeling, Studies in Fuzziness and Soft Computing, Vol. 129, Springer, 2003.

[5] B. Liu, Theory and Practice of Uncertain Programming, Studies in Fuzziness and Soft Computing, Vol. 239, Springer, 2009.

[6] 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.

[7] 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.

[8] 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.

[9] K. Kishida, H. Miyajima, M. Maeda and S. Murashima, A Self-tuning Method of Fuzzy Modeling using Vector Quantization, Proceedings of FUZZ-IEEE'97, pp397-402, 1997.

[10] 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.

[11] 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, pp.222-230, 2014.

[12] 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.

[13] S. Fukumoto, H. Miyajima, N. Shigei and K. Uchikoba, 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.

[14] 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.

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

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

[17] 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 (in print).