We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 7

Learning Algorithms for Fuzzy Inference Systems Using Vector Quantization

Hirofumi Miyajima, Noritaka Shigei and Hiromi Miyajima

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79925

Abstract

Many studies on learning of fuzzy inference systems have been made. Specifically, it is known that learning methods using vector quantization (VQ) and steepest descent method (SDM) are superior to other methods. In their learning methods, VQ is used only in determination of the initial parameters for the antecedent part of fuzzy rules. In order to improve them, some methods determining the initial parameters for the consequent part by VQ are proposed. For example, learning method composed of three stages as VQ, generalized inverse matrix (GIM), and SDM was proposed in the previous paper. In this paper, we will propose improved methods for learning process of SDM for learning methods using VQ, GIM, and SDM and show that the methods are superior in the number of rules to the conventional methods in numerical simulations.

Keywords: fuzzy inference systems, vector quantization, neural gas, generalized inverse method

1. Introduction

There have been many studies on learning of fuzzy systems [1–8]. Their aim is to construct learning methods based on SDM. Some novel methods on them have been developed which (1) generate fuzzy rules one by one starting from any number of rules, or reduce fuzzy rules one by one starting from a sufficiently large number of rules [2]; (2) use genetic algorithm (GA) and particle swarm optimization (PSO) to determine fuzzy systems [3]; (3) use fuzzy inference systems composed of a small number of input rule modules, such as single input rule modules (SIRMs) and double input rule modules (DIRMs) methods [9, 10]; and (4) use a

IntechOpen

© 2018 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

self-organization or a vector quantization technique to determine the initial assignment of parameters [11–15, 19]. Specifically, it is known that learning methods using vector quantization (VQ) and steepest descent method (SDM) are superior in the number of rules (parameters) to other methods [16, 19]. So, why is it effective to combine VQ with SDM in fuzzy modeling? First, let us explain how to combine SDM with methods other than VQ. (1) Although the learning time is short, the generation method is known to have low test accuracy, while the reduction method has high test accuracy but takes long learning time [2]. (2) The method using GA and PSO shows high accuracy when the input dimension and the number of rules are small, but it is known that there is a problem of scalability [3]. (3) SIRM and DIRM methods are excellent in scalability, but the accuracy of learning is not always sufficient [9]. As described above, many methods are not necessarily effective models because of the difficulty of learning accompanying the increase of the input dimension and the number of rules and the low accuracy. On the other hand, the method combining VQ with SDM is possible to efficiently conduct learning of SDM by arranging suitably the initial parameters of fuzzy rules using VQ [1, 16]. However, since VQ is unsupervised learning, it is easy to reflect the input part of learning data, but how to capture output information in learning is difficult. With their studies, the first learning method is the one using VQ only in determining the initial parameters of the antecedent part of fuzzy rules using input part of learning data [1, 11–14]. The second method is the one determining the same parameter using input/output parts of learning data [15, 19]. Further, the third method is one iterating learning process of VQ and SDM for the second method. Kishida and Pedrycz proposed the method based on the third one [13, 15]. These methods are the ones determining only the antecedent parameters by VQ. Therefore, we introduced generalized inverse matrix (GIM) to determine the initial assignment of weight parameters for the consequent part of fuzzy rules as the fourth method and showed the effectiveness in the previous paper [16, 17]. In this paper, improved methods for learning process of SDM in learning methods using VQ, GIM, and SDM are introduced and show that the method is superior in the number of rules to other methods in numerical simulations.

2. Preliminaries

2.1. The conventional fuzzy inference model

The conventional fuzzy inference model using SDM is described [1]. Let $Z_j = \{1, \ldots, j\}$ and $Zj* = \{0, 1, \ldots, j\}$. Let R be the set of real numbers. Let $\boldsymbol{x} = (x_1, \ldots, x_m)$ and y be input and output variables, respectively, where $x_j \in R$ for $j \in Z_m$, and $y \in R$. Then, the rule of simplified fuzzy inference model is expressed as

$$R_i: \text{if } x_1 \text{ is } Mi_1 \text{ and } x_j \text{ is } M_{ij} \cdot \text{ and } x_m \text{ is } M_{im}, \text{ then } y \text{ is } w_i \tag{1}$$

where $j \in Z_m$ is a rule number, $i \in Z_n$ 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

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

Then, the output y^* of fuzzy inference method is obtained as

$$y^* = \frac{\sum_{i=1}^{n} \mu_i \cdot w_i}{\sum_{i=1}^{n} \mu_i} \tag{3}$$

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

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

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

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

Let $D = \{(x^p, \ldots, x^p, y^r) \mid p \in Z_P\}$ and $D^* = \{(x^p, \ldots, x^p) \mid p \in Z_p\}$ be the set of learning data and the set of input part of D, respectively. The objective of learning is to minimize the following mean square error (MSE) as

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

where $yp*$ and y^r mean inference and desired output for the pth input $\boldsymbol{x}^p$.

In order to minimize the objective function E, each parameter of $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ is updated based on SDM using the following relation:

$$\frac{\partial E}{\partial w_i} = \frac{\mu_i}{\sum_{I=1}^{n} \mu_I} \cdot (y^* - y^r) \tag{6}$$

$$\frac{\partial E}{\partial c_{ij}} = \frac{\mu_i}{\sum_{I=1}^{n} \mu_I} \cdot (y^* - y^r) \cdot (w_i - y^*) \cdot \frac{x_j - c_{ij}}{b_{ij}^2} \tag{7}$$

$$\frac{\partial E}{\partial c_{ij}} = \frac{\mu_i}{\sum_{I=1}^{n} \mu_I} \cdot (y^* - y^r) \cdot (w_i - y^*) \cdot \frac{\left(x_j - c_{ij}\right)^2}{b_{ij}^3} \tag{8}$$

where t is iteration time and K_α is a learning constant [1].

The learning algorithm for the conventional fuzzy inference model is shown as follows:

Learning Algorithm A

Step A1: The threshold θ of inference error and the maximum number of learning time T_{max} are set. Let n_0 be the initial number of rules. Let $t = 1$.

Step A2: The parameters b_{ij}, c_{ij}, and w_i are set randomly.

Step A3: Let $p = 1$.

Step A4: A data $\left(x_1^p, \cdots, x_m^p, y_p^r\right) \in D$ is given.

Step A5: From Eqs. (2) and (3), μ_i and y^* are computed.

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

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 $E(t) \leq \theta$ and $t \leq T_{max}$, then the algorithm terminates.

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

In particular, Algorithm SDM is defined as follows:

Algorithm SDM (*c*, *b*, *w*)

θ_1: inference error

T_{max1}: the maximum number of learning time

n: the number of rules

input: current parameters

output: parameters $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ after learning

Steps A3 to A8 of Algorithm A are performed.

2.2. Neural gas method

Vector quantization techniques encode a data space $V \subseteq R^m$, utilizing only a finite set $C = \{\boldsymbol{c}_i | i \in Z_r\}$ of reference vectors [18].

Let the winner vector $\boldsymbol{c}_i(\boldsymbol{v})$ be defined for any vector $\boldsymbol{v} \in \mathbf{V}$ as

$$i(\boldsymbol{v}) = \arg\min_{i \in Z_r} \|\boldsymbol{v} - c_i\|. \tag{9}$$

By using the finite set C, the space V is partitioned as

$$V_i = \left\{\boldsymbol{v} \in V \,|\, \|\boldsymbol{v} - \boldsymbol{c}_i\| \leq \|\boldsymbol{v} - \boldsymbol{c}_j\| \text{ f or } j \in \mathrm{Z}_r\right\}, \tag{10}$$

where $V = \cup_{i \in Z_r} V_i$ and $V_i \cap V_j = \varphi$ for $i \neq j$.

http://dx.doi.org/10.5772/intechopen.79925

The evaluation function for the partition is defined by

$$E = \sum_{i=1}^{r} \sum_{v \in V_i} \frac{1}{n_i} \|v - c_i\|^2, \tag{11}$$

where $n_i = |V_i|$.

Let us introduce the neural gas method as follows [18]:

For any input data vector v, the neighborhood ranking $c_i k$ for $k \in Z_{r-1}^*$ is

determined, being the reference vector for which there are k vectors c_j with

$$\|v - c_j\| < \|v - c_i k\| \tag{12}$$

Let the number k associated with each vector c_i denoted by $k_i(v,c_i)$. Then, the adaption step for adjusting the parameters is given by

$$\triangle c_i = \varepsilon \cdot h_\lambda(k_i(v, c_i)) \cdot (v - c_i) \tag{13}$$

$$h_\lambda(k_i(v, c_i)) = \exp(-k_i(v, c_i)/\lambda) \tag{14}$$

where $\varepsilon \in [0, 1]$ and $\lambda > 0$.

Let the probability of v selected from V be denoted by $p(v)$.

The flowchart of the conventional neural gas algorithm is shown in **Figure 1** [18], where ε_{int}, ε_{fin}, and T_{max2} are learning constants and the maximum number of learning, respectively. The method is called learning algorithm NG.

Using the set D, a decision procedure for center and width parameters is given as follows:

Algorithm Center (c)

$D^* = \{(x^p, \ldots, x^p) | p \in Z_p\}$

$p(x)$: the probability of x selected for $x \in D^*$.

Step 1: By using $p(x)$ for $x \in D^*$, NG method of **Figure 1** [16, 18] is performed.

As a result, the set C of reference vectors for D^* is determined, where $C = n$.

Step 2: Each value for center parameters is assigned to a reference vector. Let

$$b_{ij} = \frac{1}{n_i} \sum_{x_k \in C_i} \left(c_{ij} - x_{kj}\right)^2 \tag{15}$$

where C_i and n_i are the set and the number of learning data belonging to the ith cluster C_i and $C = \cup_{i=1}^{r} C_i$ and $n = \sum_{i=1}^{r} n_i$.

As a result, center and width parameters are determined from algorithm center (c).

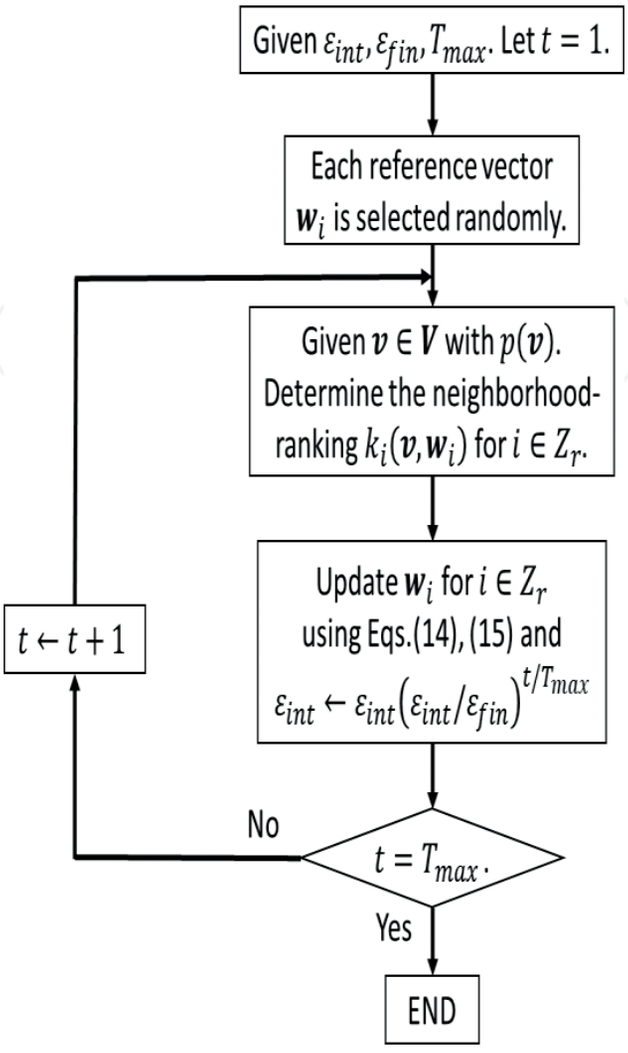


Figure 1. Neural gas method [18].

Learning Algorithm B using Algorithm Center ($\boldsymbol{c}$) is introduced as follows [16, 17]:

Learning Algorithm B

θ: threshold of MSE

T_{max}^{0}: maximum number of learning time for NG

T_{max}: maximum number of learning time for SDM

M: the size of ranges

n: the number of rules

Step 1: Initialize()

Step 2: Center and width parameters are determined from Algorithm Center(P) and the set D^{*}.

Step 3: Parameters $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ are updated using Algorithm SDM ($\boldsymbol{c}$, $\boldsymbol{b}$, $\boldsymbol{w}$).

Step 4: If $E(t) \leq \theta$, then algorithm terminates else go to Step 3 with $n \leftarrow n + 1$ and $t \leftarrow t + 1$.

http://dx.doi.org/10.5772/intechopen.79925

2.3. The probability distribution of input data based on the rate of change of output

It is known that many rules are needed at or near the places where output data change quickly in fuzzy modeling. Then, how can we find the rate of output change? The probability $p_M(\boldsymbol{x})$ is one method to perform it. As shown in Eqs. (16) and (17), any input data where output changes quickly is selected with the high probability, and any input data where output changes slowly is selected with the low probability, where M is the size of range considering output change.

Based on the literature [13], the probability (distribution) is defined as follows:

Algorithm Prob ($p_M(\boldsymbol{x})$)

Input: $D = \{(\boldsymbol{x}^p, y^r) \mid p \in Z_P\}$ and $D^* = \{(\boldsymbol{x}^p) \mid p \in Z_P\}$

Output: $p_M(\boldsymbol{x})$

Step 1: Give an input data $\boldsymbol{x}^i \in D^*$, we determine the neighborhood ranking ($\boldsymbol{x}^i 0$, $\boldsymbol{x}^i 1$, …, $\boldsymbol{x}^i k$,…, $\boldsymbol{x}^i P-1$) 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 $\boldsymbol{x}^i k$ ($k = 0, \ldots, P-1$) being the vector $\boldsymbol{x}^i$ for which there are k vectors $\boldsymbol{x}^j$ with $\left\|\boldsymbol{x}^i - \boldsymbol{x}^j\right\| < \left\|\boldsymbol{x}^i - \boldsymbol{x}^{i_k}\right\|$.

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

$$H(\boldsymbol{x}^i) = \sum_{l=1}^{M} \frac{\left|y^i - y^{i_l}\right|}{\left\|\boldsymbol{x}^i - \boldsymbol{x}^{i_l}\right\|}, \tag{16}$$

where $\boldsymbol{x}^{i_l}$ for l Z_M means the lth 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 probability $p_M(\boldsymbol{x}^i)$ for $\boldsymbol{x}^i$ by normalizing $H(\boldsymbol{x}^i)$ as follows:

$$p_M(\boldsymbol{x}^i) = \frac{H(\boldsymbol{x}^i)}{\sum_{j=1}^{P} H(\boldsymbol{x}^j)}, \tag{17}$$

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

See Ref. [19] for the detailed explanation using the example of $p_M(\boldsymbol{x})$. Using $p_M(\boldsymbol{x})$, Kishida has proposed the following learning algorithm [13]:

Learning Algorithm C

θ: threshold of MSE

T_{max}^0: maximum number of learning time for NG

T_{max}: maximum number of learning time for SDM

M: the size of ranges

n: the number of rules

Step 1: Initialize ( )

Step 2: The probability $p_M(\boldsymbol{x})$ is obtained from algorithm prob ($p_M(\boldsymbol{x})$).

Step 3: Center and width parameters are determined using $p_M(\boldsymbol{x})$ from Algorithm Center (P) and the data set D.

Step 4: Parameters $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ are updated using Algorithm SDM ($\boldsymbol{c}$, $\boldsymbol{b}$, $\boldsymbol{w}$).

Step 5: If $E(t) \leq \theta$, then algorithm terminates else go to Step 3 with $n \leftarrow n + 1$ and t = 1.

2.4. Determination of weight parameters using the generalized inverse method

The optimum values of parameters $\boldsymbol{c}$ and $\boldsymbol{b}$ are determined by using $p_K(\boldsymbol{x})$. Then, how can we decide weight parameters $\boldsymbol{w}$? We can determine them as the interpolation problem for parameters $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$. That is, it is the method that membership values for antecedent part of rules are computed from $\boldsymbol{c}$ and $\boldsymbol{b}$ and weight parameters $\boldsymbol{w}$ are determined by solving the interpolation problem. So far, the method was used as a determination problem of weight parameters for RBF networks [1].

Let us explain fuzzy inference systems and interpolation problem using the generalized inverse method [1]. This problem can be stated mathematically as follows:

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

$$f(\boldsymbol{x}^p) = y_p^r \tag{18}$$

In fuzzy modeling, this problem is solved as follows:

$$y_p = f(\boldsymbol{x}^p) = \sum_{i=1}^{n} w_i \varphi_{pi}(\|\boldsymbol{x}^p - \boldsymbol{c}_i\|) \tag{19}$$

$$\varphi_{pi}(\|\boldsymbol{x}^p - \boldsymbol{c}_i\|) = \frac{\mu_i}{\sum_{I=1}^{n} \mu_I}, \mu_i = \prod_{j=1}^{m} M_{ij}(x_j), \tag{20}$$

where μ_i and M_{ij} are defined as Eqs. (2) and (4).

That is,

$$\varphi \, \mathbf{w} = \mathbf{y}, \tag{21}$$

where $\varphi = (\varphi_{ij})$ ($i \in Z_P$ and $j \in Z_n$), $\boldsymbol{w} = (w_1, \ldots, w_n)^T$, and $\boldsymbol{y} = (y_1^r, \ldots, y_p^r)^T$.

Let $P = n$ and $\boldsymbol{x}^i = \boldsymbol{c}_i$. The width parameters are determined by Eq. (15). Then, if $\varphi_{ij}( )$ is suitably selected as Gaussian function, then the solution of weights $\boldsymbol{w}$ is obtained as

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

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

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

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

The matrix Φ^+ is called the generalized inverse of φ. The method using Φ^+ to determine the weights is called the generalized inverse method (GIM).

Using GIM, a decision procedure for parameters is defined as follows:

Algorithm Weight(*c*, *b*)

Input: $D = \{(\boldsymbol{x}^p, y^r) | p \in Z_P\}$

Output: The weight parameters $\boldsymbol{w}$

Step 1: Calculate μ_i based on Eq. (2)

Step 2: Calculate the matrix Φ and Φ^+ using Eq. (20):

$$\varphi_{pi} = (\|x^p - c_i\|) = \frac{\mu_i^p}{\sum_{j=1}^n \mu_j^p}, \mu_i^p = \prod_{j=1}^m \exp\left(-\frac{1}{2}\left(\frac{x_j^p - c_{ij}}{b_{ij}}\right)^2\right)$$

Step 3: Determine the weight vectors $\boldsymbol{w}$ as follows:

$$\boldsymbol{w} = \Phi^+ \boldsymbol{y}^r \tag{24}$$

2.5. The relation between the proposed algorithm and related works

Let us explain the relation between the proposed method and related works using **Figure 2**.

1. The fundamental flow of algorithm A is shown in **Figure 2**(**a**). Initial parameters of $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$ are set randomly, and all parameters are updated using SDM until the inference error become sufficiently small (see **Figure 2**(**a**)) [1].
2. The first method using VQ is the one that both the initial assignment of parameters and the assignment of parameters in iterating step (see outer loop of **Figure 2**(**b**)) are also determined by NG using D^*. That is, it is learning method composed of two stages. The center parameters $\boldsymbol{c}$ are determined using D^* by VQ, $\boldsymbol{b}$ is computed by Eq. (15) using the result of center parameters, and weight parameter $\boldsymbol{w}$ 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. In iterating processes, parameters of the result obtained by SDM are set as initial ones of the next process. Outer iterating process is repeated until the inference error become sufficiently small (see **Figure 2**(**b**)).
3. The second method using VQ is the one that is the same method as the first one except for selecting any learning data based on $p_M(\boldsymbol{x})$ (see **Figure 2**(**c**)). That is, center parameters $\boldsymbol{c}$ are determined by $p_M(\boldsymbol{x})$ using input and output learning data.

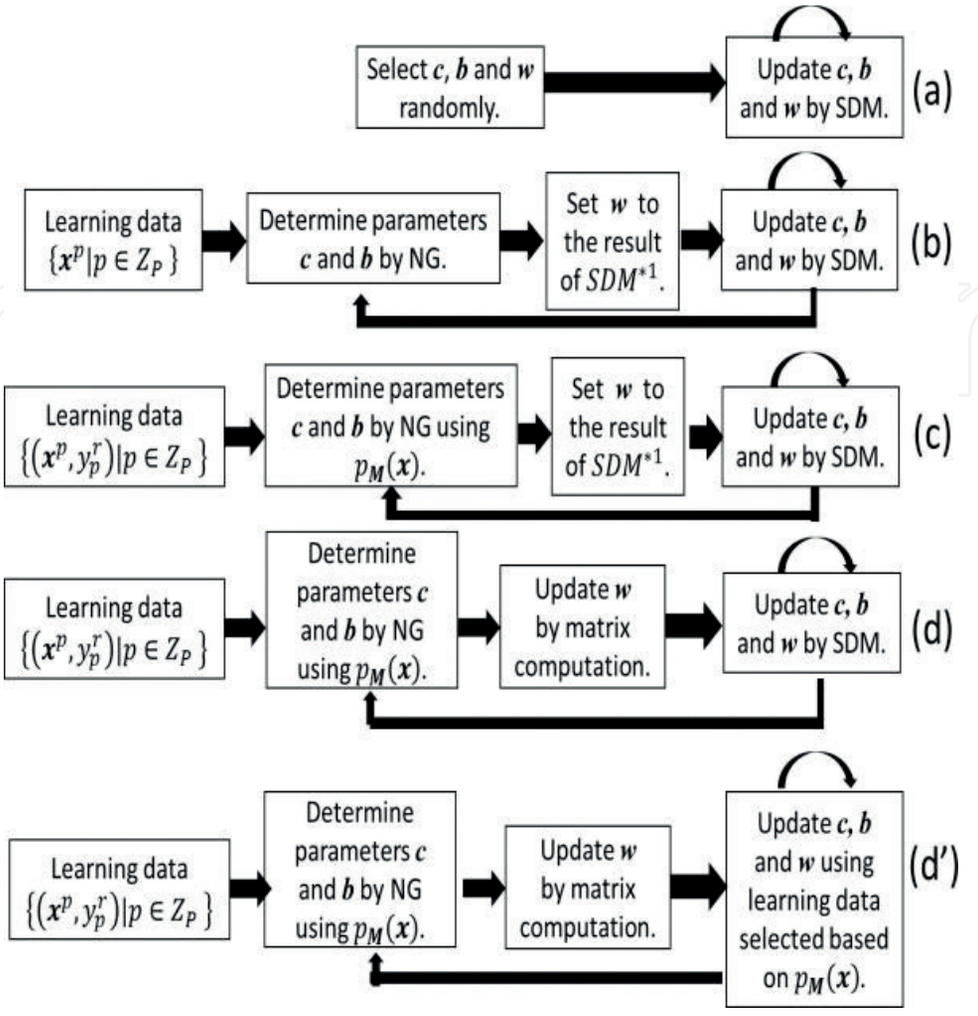


Figure 2. Concept of conventional and proposed algorithms: mark 1 means that initial values of w are selected randomly and parameters w are set to the result of SDM after the second step.

4. The third learning method using VQ is the one that parameters w are determined using GIM after parameters c and b are determined by VQ using $p_M(x)$ and all parameters are updated based on SDM. That is, it is learning method composed of three phases. In the first phase, the center parameters c are determined using the probability $p_M(x)$, and b is computed from the result of center parameters. In the second phase, weight parameters w are determined by solving the interpolation problem using GIM. In the third phase, all parameters are updated using SDM for the definite number of learning time. In iterating process, the result of SDM is set to initial ones of the next process based on hill climing. Outer process is repeated until the inference error becomes sufficiently small (see **Figure 2**(**d**)).

5. The fourth method is the same to the one as the third method except for using $p_M(x)$ in learning process of SDM (see **Figure 2**(**d'**)). This is a proposed method in this paper.

http://dx.doi.org/10.5772/intechopen.79925

3. The proposed learning method using VQ

Let us explain the detailed algorithm of **Figure 2**(**d′**). The method is called Learning Algorithm D′. It is composed of four techniques as follows:

1. Determine the initial assignment of $\boldsymbol{c}$ using the probability $p_K(\boldsymbol{x})$.
2. Determine the assignment of weight parameters $\boldsymbol{w}$ by solving the interpolation problem using GIM.
3. The processes (1) and (2) and learning steps of SDM using $p_M(\boldsymbol{x})$ are iterated.
4. The optimum value of M is determined by hill climing method [16].

The general scheme of the proposed method is shown as **Figure 3**, where $\boldsymbol{c}_{min}$, $\boldsymbol{b}_{min}$, and $\boldsymbol{w}_{min}$ are the optimal parameters for $\boldsymbol{c}$, $\boldsymbol{b}$, and $\boldsymbol{w}$.

T_{max1} and T_{max2}: The maximum numbers of learning time for NG and SDM.

θ and θ_1: Thresholds for MSE and SDM

M_0, M_{max}: The size of initial and final of ranges

$\triangle M$: The rate of change of the range

D and D^*: Learning data $D = \{(\boldsymbol{x}^i, y^r) \mid i \in Z_P\}$ and $D^* = \{\boldsymbol{x}^i \mid i \in Z_P\}$

n: The number of rules

$E(t)$: MSE of inference error at step t

E_{min}: The minimum MSE of E for the rule number

The proposed method of **Figure 3** consists of five phases: In the first phase, all values for algorithm are initialized. In the second phase, the probability $p_M(\boldsymbol{x})$ is determined for the size of range M. In the third phase, parameters $\boldsymbol{c}$ are determined by NG using $p_M(\boldsymbol{x})$, and parameters $\boldsymbol{b}$ are computed from parameters $\boldsymbol{c}$. In the forth phase, parameters $\boldsymbol{w}$ are determined from algorithm weight($\boldsymbol{c}$, $\boldsymbol{b}$). In the fifth phase, all parameters are updated using $p_M(\boldsymbol{x})$ by SDM. The optimum number n^* of rules and the optimum size M^* of range are determined in **Figure 4**. That is, the number M for the fixed number n is adjusted, and the optimum values of n^* and M^* with the minimum number for MSE are determined. Especially, Learning Algorithm D is same method as Learning Algorithm D′ except for the step with the symbol "*" in **Figure 3**. In learning steps of SDM for Learning Algorithm D, learning data is selected randomly (see **Figure 2**(**d**)).

Likewise, we also propose improved methods for **Figure 2**(**a**)–(**c**). In learning process of SDM for algorithm (a), (b), and (c), any learning data is selected randomly. In the proposed methods, any learning data is selected based on $p_M(\boldsymbol{x})$. These algorithms are defined as (a′), (b′), and (c′).

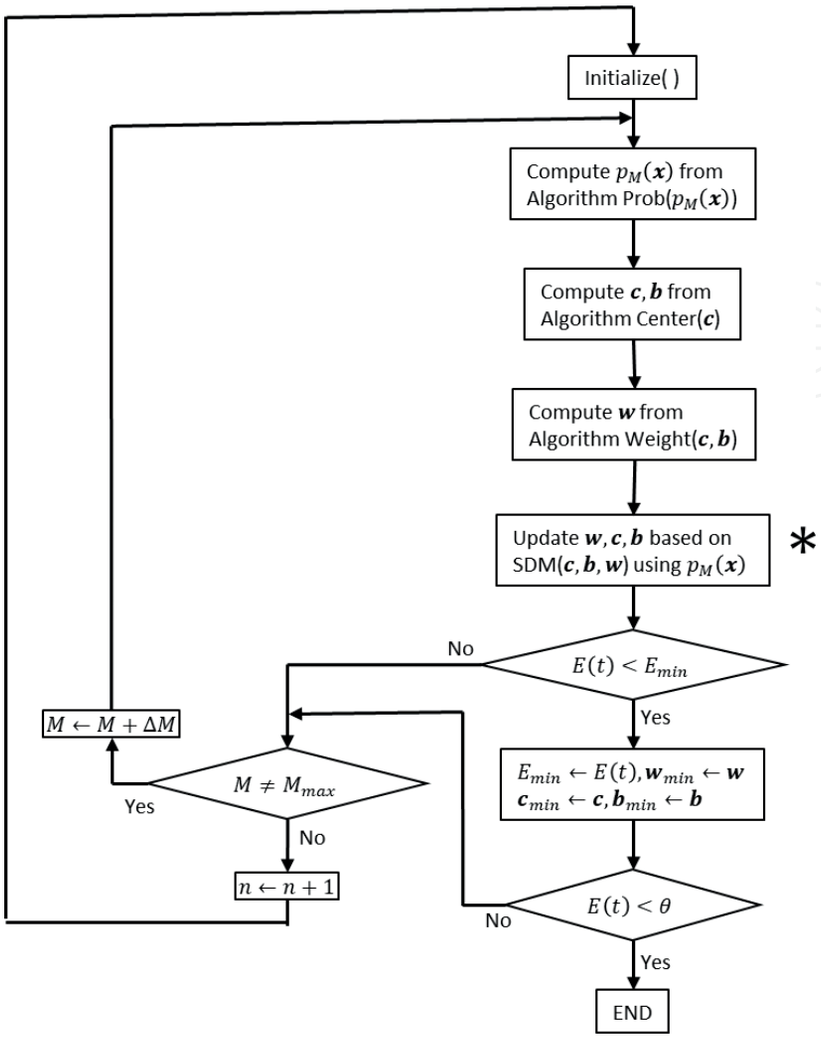


Figure 3. Flowchart of Learning Algorithm D′ corresponding to **Figure 2(d′)**.

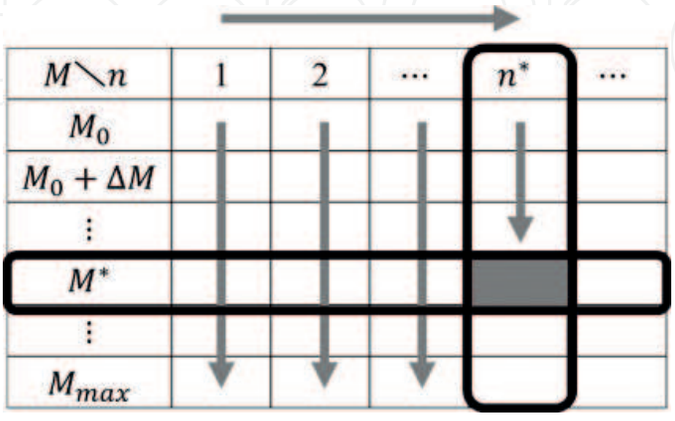


Figure 4. The optimum values M^* and n^* for M and n.

http://dx.doi.org/10.5772/intechopen.79925

4. Numerical simulations

In order to compare the ability of Learning Algorithms (a′), (b′), (c′), and (d′) with Learning Algorithms (a), (b), (c), and (d), numerical simulations for function approximation and pattern classification are performed.

4.1. Function approximation

The systems are identified by fuzzy inference systems. This simulation uses four systems specified by the following functions with two-dimensional input space $[0, 1]^2$ (Eqs. (25)–(28)) and one output with the range [0, 1];

$$y = \sin\left(\pi x_1^3\right) x_2 \tag{25}$$

$$y = \frac{\sin\left(2\pi x_1^3\right)\cos\left(\pi x_2\right) + 1}{2} \tag{26}$$

$$y = \frac{1.9\left(\left(1.35 + \exp\left(x_1\right)\right)\sin\left(13(x_1 - 0.6)^2\right)\exp\left(-x_2\right)\sin\left(7x_2\right)\right)}{2} \tag{27}$$

$$y = \frac{\sin\left(10(x_1 - 0.5)^2 + 10(x_2 - 0.5)^2\right) + 1}{2} \tag{28}$$

In this simulation, $T_{max1} = 100000$ and $T_{max2} = 50000$ for (a) and $T_{max1} = 10000$ and $T_{max2} = 5000$ for (b), (c), and (d) and $\theta = 1.0 \times 10^{-4}$, $K_0 = 100$, $K_{max} = 190$, $K = 10$, $K_c = 0.01$, $K_b = 0.01$, $K_c = 0.1$, the number of learning data is 200 and the number of test data is 2500.

Table 1 shows the results for the simulation. In **Table 1**, the number of rules, MSEs for learning and test, and learning time (second) are shown, where the number of rules means the one when the threshold θ of inference error is achieved in learning. The result of simulation is the average value from 20 trials. As a result, the results of (a′), (b′), (c′), and (d′) are almost same as the cases of (a), (b), (c), and (d) as shown in **Table 1**. It seems that there is no difference of the ability for the regression problem.

4.2. Classification problems for UCI database

Iris, Wine, Sonar, and BCW data from UCI database shown in **Table 2** are used as the second numerical simulation [20]. In this simulation, fivefold cross validation is used. As the initial conditions for classification problem, $K_c = 0.001$, $K_b = 0.001$, $K_w = 0.05$, $\varepsilon_{init} = 0.1$, $\varepsilon_{fin} = 0.01$, and $\lambda = 0.7$ are used. Further, $T_{max} = 50000$, $M = 100$, and $\theta = 1.0 \times 10^{-2}$ for iris and wine. $T_{max} = 50000$, $M = 200$, and $\theta = 2.0 \times 10^{-2}$ for BCW; and $T_{max} = 5000$, $M = 100$, and $\theta = 5.0 \times 10^{-2}$ for sonar are used.

Table 3 shows the result of classification problem. In **Table 3**, the number of rules, RMs for learning, and test data are shown, where RM means the rate of misclassification. As a result, the

		Eq. (25)	Eq. (26)	Eq. (27)	Eq. (28)
(a)	The number of rules	8.3	22.5	52.4	6.1
	MSE for learning($\times 10^{-4}$)	0.47	0.35	0.65	0.41
	MSE of test ($\times 10^{-4}$)	2.29	21.12	2.83	7.37
(b)	The number of rules	4.7	6.8	9.6	4.0
	MSE of learning ($\times 10^{-4}$)	0.44	0.38	0.84	0.35
	MSE of test ($\times 10^{-4}$)	0.70	2.96	2.34	0.48
(c)	The number of rules	5.4	7.4	11.1	3.5
	MSE of learning ($\times 10^{-4}$)	0.24	0.54	0.65	0.33
	MSE of test ($\times 10^{-4}$)	0.65	1.36	4.48	0.44
(d)	The number of rules	4.3	6.1	9.7	3.5
	MSE of learning ($\times 10^{-4}$)	0.28	0.39	0.69	0.29
	MSE of test ($\times 10^{-4}$)	0.57	1.93	1.78	0.36
(a′)	The number of rules	5.0	8.9	11.8	4.7
	MSE for learning ($\times 10^{-4}$)	0.37	0.41	0.52	0.45
	MSE of test ($\times 10^{-4}$)	1.55	9.56	2.8	1.06
(b′)	The number of rules	5.0	8.9	13.0	4.3
	MSE for learning ($\times 10^{-4}$)	0.42	0.38	0.65	0.39
	MSE of test ($\times 10^{-4}$)	1.41	9.66	4.12	2.38
(c′)	The number of rules	5.7	8.0	13.1	4.1
	MSE for learning ($\times 10^{-4}$)	0.40	0.23	0.57	0.35
	MSE of test ($\times 10^{-4}$)	1.70	1.28	3.90	1.10
(d′)	The number of rules	4.6	6.9	10.0	3.6
	MSE for learning ($\times 10^{-4}$)	0.39	0.49	0.62	0.35
	MSE of test ($\times 10^{-4}$)	1.43	2.58	1.89	0.42

Table 1. The results of simulations for function approximation.

	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 2. The dataset for pattern classification.

results of (a′), (b′), (c′), and (d′) are superior in the number of rules to the cases of (a), (b), (c), and (d) as shown in **Table 3**. It seems that there is the difference of ability for pattern classification.

Let us consider the reason why we can get the good result by using the probability $p_M(\boldsymbol{x})$. In the conventional learning method, parameters are updated by any data selected randomly

http://dx.doi.org/10.5772/intechopen.79925

		Iris	Wine	BCW	Sonar
(a)	The number of rules	3.4	7.8	14.4	11.0
	RM for learning (%)	3.0	1.4	1.6	5.3
	RM of test (%)	3.3	10.3	4.3	20.6
(b)	The number of rules	2.0	20.8	26.0	3.7
	RM of learning (%)	3.3	13.6	2.2	5.1
	RM of test (%)	3.3	16.6	3.5	18.2
(c)	The number of rules	2.0	3.2	4.8	4.0
	RM of learning (%)	3.3	1.5	1.6	5.1
	RM of test (%)	4.0	6.7	3.8	19.0
(d)	The number of rules	3.7	2.5	2.5	4.0
	RM of learning (%)	3.3	1.1	1.3	5.1
	RM of test (%)	3.8	6.5	2.1	18.3
(a′)	The number of rules	2.3	2.2	3.5	4.6
	RM for learning (%)	2.9	1.4	1.6	5.0
	RM of test (%)	3.5	8.5	3.9	20.0
(b′)	The number of rules	2.0	2.0	2.1	3.7
	RM for learning (%)	3.9	3.0	2.1	5.0
	RM of test (%)	4.9	9.2	3.9	19.0
(c′)	The number of rules	2.3	3.0	3.6	4.0
	RM for learning (%)	3.3	2.6	2.2	5.3
	RM of test (%)	4.0	7.2	3.5	19.4
(d′)	The number of rules	2.3	2.0	2.4	3.3
	RM for learning (%)	3.0	1.8	2.2	5.0
	RM of test (%)	3.5	7.6	3.7	19.1

Table 3. The result for pattern classification.

from the set of learning data. In the proposed method, parameters are updated by any data selected based on the probability $p_M(\boldsymbol{x})$. The probability $p_M(\boldsymbol{x})$ is determined based on output change for learning data, so many fuzzy rules are likely to generate at or near the places where output change is large for the set of learning data.

For example, if the number of learning time is 100 and $p_M(\boldsymbol{x}^0) = 0.5$, then learning data $\boldsymbol{x}^0$ is selected 50 times from the set of learning data in learning. As a result, membership functions are likely to generate at or near the places where output change is large for the set of learning data. The probability $p_M(\boldsymbol{x})$ is used in a method to improve the local search of SDM.

5. Conclusion

In this paper, we proposed the improved methods using VQ, GIM, and SDM. The features of the proposed methods are as follows:

1. In determining the initial assignment of parameters, both input and output parts of learning data are used.
2. The initial assignment of weight parameters is determined by GIM.
3. In order to determine the range of the rate of output change, hill climing is used.
4. Any learning data in SDM is selected based on the probability distribution

$p_M(\boldsymbol{x})$ considering both input and output of learning data.

As a result, it was shown that the proposed methods using the probability distribution considering both input and output parts of learning data were superior to other methods in numerical simulation of pattern classification.

In the future works, we will consider the new idea using VQ and apply the proposed method to control problem.

Author details

Hirofumi Miyajima[1], Noritaka Shigei[2] and Hiromi Miyajima[3]*

*Address all correspondence to: k2356323@kadai.jp

1 Faculty of Informatics, Okayama University of Science, Okayama, Japan

2 Kagoshima University, Kagoshima, Japan

3 Former Kagoshima University, Kagoshima, Japan

References

[1] Gupta MM et al. Static and Dynamic Neural Networks. John Wiley & Sons: IEEE Press; 2004

[2] Fukumoto S, Miyajima H, Kishida K, Nagasawa Y. A Destructive Learning Method of Fuzzy Inference Rules. Proc. of IEEE on Fuzzy Systems;1995. pp. 687-694

[3] Cordon O. A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems, designing interpretable genetic fuzzy systems. Journal of Approximate Reasoning. 2011;**52**:894-913

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

[5] Lin C, Lee C. Neural Fuzzy Systems. PTR: Prentice Hall; 1996

[6] Casillas J, Cordon O, Herrera F, Magdalena L. Accuracy Improvements in Linguistic Fuzzy Modeling, Studies in Fuzziness and Soft Computing. Vol. 129. Berlin Heidelberg: Springer-Verlag; 2003

[7] Liu B. Theory and Practice of Uncertain Programming, Studies in Fuzziness and Soft Computing. Vol. 239. Physica-Verlag Heidelberg: Springer; 2009

[8] Zhoua SM, Ganb JQ. Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modeling. Fuzzy Sets and Systems. 2008; **159**:3091-3131

[9] Miyajima H et al. SIRMs fuzzy inference model with linear transformation of input variables and universal approximation, advances in computational intelligence. Proc. 13th International Work Conference on Artificial Neural Networks, Part I. pp. 561-575, Spain; 2015

[10] Yubazaki N, Yi J, Hirota K. SIRMS (single input rule modules) connected fuzzy inference model. Journal Advanced Computational Intelligence. 1997;**1**(1):23-30

[11] Kishida K et al. A self-tuning method of fuzzy modeling using vector quantization. Proc. FUZZ-IEEE'97. pp. 397-402;1997

[12] Kishida K et al. Destructive fuzzy modeling using neural gas network. IEICE Trans. on Fundamentals. 1997;**E80-A**(9):1578-1584

[13] Kishida K et al. A learning method of fuzzy inference rules using vector quantization. Proceedings of the 21st International Conference on Artificial Neural Networks. 1998;**2**: 827-832

[14] Fukumoto S et al. A decision procedure of the initial values of fuzzy inference system using counterpropagation networks. Journal of Signal Processing. 2005;**9**(4):335-342

[15] Pedrycz W et al. Cluster-centric fuzzy modeling. IEEE Transactions on Fuzzy Systems. 2014;**22**(6):1585-1597

[16] Miyajima H et al. Fast learning algorithm for fuzzy inference systems using vector quantization. International MultiConference of Engineers and Computer Scientists. 2016;**I**:1-6

[17] Miyajima H et al. The ability of learning algorithms for fuzzy inference systems using vector quantization. ICONIP 2016, part IV, LNCS9950; 2016. pp. 479-488

[18] Martinetz TM et al. Neural gas network for vector quantization and its application to time-series prediction. IEEE Transaction on Neural Network. 1993;**4**:558-569

[19] Miyajima H et al. An improved learning algorithm of fuzzy inference systems using vector quantization. Advanced in Fuzzy Sets and Systems. 2016;**21**(1):59-77

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

IntechOpen

IntechOpen