PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 37 (2004) 2049–2057

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Adaptive fuzzy inference neural network

Hitoshi Iyatomi*, Masafumi Hagiwara

Department of Information and Computer Science, Keio University, 3-14-1 Hiyoshi, Yokohama 223-8522, Japan

Received 6 February 2003; received in revised form 5 April 2004; accepted 5 April 2004

Abstract

An adaptive fuzzy inference neural network (AFINN) is proposed in this paper. It has self-construction ability, parameter estimation ability and rule extraction ability. The structure of AFINN is formed by the following four phases: (1) initial rule creation, (2) selection of important input elements, (3) identification of the network structure and (4) parameter estimation using LMS (least-mean square) algorithm. When the number of input dimension is large, the conventional fuzzy systems often cannot handle the task correctly because the degree of each rule becomes too small. AFINN solves such a problem by modification of the learning and inference algorithm.

Keywords: Neural network; Fuzzy inference; Machine learning; Fuzzy modeling and rule extraction

1. Introduction

Numeric analysis approach of fuzzy system was first presented by Takagi and Sugeno [1] and then a lot of studies have been made [1–11]. Since the systems using fuzzy theory can express rules or knowledge as “if-then” form, they have advantages such as they do not need mathematical analysis for modeling. However, they need the appropriate model construction and parameter selection [1–7]. This kind of fuzzy modeling problem is a troublesome work in general.

On the other hand, studies of fuzzy neural networks that combine both advantages of the fuzzy systems and the learning ability of the neural networks have been carried out. These techniques can alleviate the matter of fuzzy modeling by learning ability of neural networks and have been reported since around the beginning of 1990s [2,3]. Fuzzy neural networks can be applied not only for simple pattern classification but also meaningful fuzzy if-then rules creation; therefore, they can be put into practice for various applications. In the early stage of fuzzy neural network researches, Lin et al. [2] proposed one of the current prima models which decide the initial fuzzy model by Kohonen’s self-organizing algorithm [8] and carry out parameter adjustment by back propagation algorithm. Also as a representative example, Jang et al. proposed ANFIS [9] in 1993. ANFIS applies a neural network in determination of the shape of membership functions and rule extraction. However, since it needs to divide the input data space in advance, accuracy of the system depends much on the achievement of this pre-processing. Wang et al. [10] reported an approach to acquire fuzzy rules by dividing input space. These techniques, however, do not consider the output data space, so the obtained rules should not be always reasonable. Nishina et al. proposed fuzzy inference neural network (FINN) [11]. FINN is a simple and effective fuzzy neural network that divides input–output data space and extracts fuzzy if-then rules automatically. Likewise many other fuzzy neural networks, FINN decides initial rules by self-organizing learning at first, then carries out merging rules based on Euclidean distance and uses LMS (least-mean square) algorithm

* Corresponding author. Tel.: +81-45-566-1762; fax: +81-45-566-1747.
E-mail address: iyatomi@soft.ics.keio.ac.jp (H. Iyatomi)

0031-3203/$ - see front matter © 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.04.003

to adjust the parameters. Since the architecture and behavior of FINN are very applicable, it has been adopted as a basic component for image recognition and interpretation researches [12,13]. However, its fuzzy modeling for the target task is not always sufficient.

A lot of systems which aim at excellent fuzzy modeling and carry out input selection, rule creation and parameter estimation have been proposed [4–7,14,15]. Elimination of unnecessary rules and selection of efficient input elements can contribute the performance improvement, reduction of calculation cost and analysis of the obtained rules that is one of the most important merits of fuzzy systems. In these researches, Linkens et al. [6,7] reported effective input selection and rule creation method and showed the excellent results on their experiments. However, since these methods use fuzzy c-means (FCM) based algorithm as well as [16,17], they have to know the number of proper rules in advance. In addition, since the algorithms are complicated and their algorithms and results are presented only for single output system, they have difficulty if they are employed for many applications.

Generally, many fuzzy neural network models have common problems derived from their fundamental algorithm. For example, the systems which use gradient decent method sometimes reduce the width of the membership function to negative during the learning. The systems which employ basic fuzzy inference theory make the degree of each rule extremely small and often make it underflow when the dimension of the task is large. In such a situation, the learning and inference cannot be carried out correctly.

In this paper, we propose a new adaptive fuzzy inference neural network (AFINN) that alleviates these shortcomings of the conventional models. AFINN carries out appropriate model construction and solves fuzzy-neuro specific problems by modification of the fuzzy inference and learning algorithm.

Following four steps perform model construction of the AFINN:

1. determination of the initial rules by adaptive self-organizing learning;
2. selection of important input elements;
3. determination of the fuzzy neural network model by adaptive self-organizing learning;
4. parameter estimation using LMS learning.

In Section 2, structure and behavior of the proposed AFINN are described. In Section 3, the model construction of AFINN including input selection, rule creation and parameter estimation are explained. In Section 4, three examples are used to examine the efficiency of AFINN. Conclusions are summarized in the last section.

2. AFINN

To construct appropriate fuzzy model, the following problems should be solved:

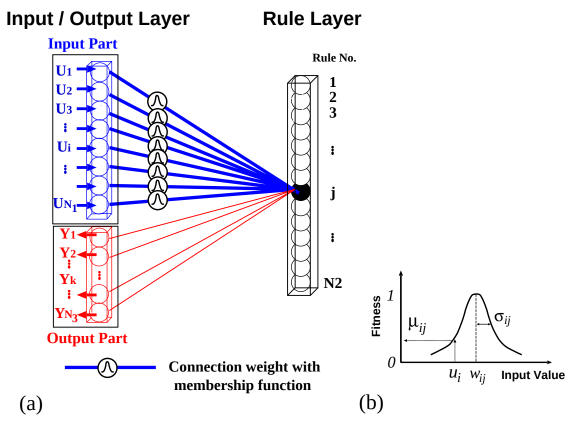

Fig. 1. (a) Structure of the AFINN and (b) its membership function.

1. identification of the optimum number of rules;
2. identification of the optimum element of inputs;
3. identification of the system parameter.

The proposed AFINN aims for the realization of above features and performs fuzzy-modeling based on modeling parameters such as ξ_{self} and ε_{self}, those are defined later. The modeling scheme is designed with the simple and versatile FINN architecture [11]. First, we explain the structure and the behavior of AFINN.

2.1. Structure of AFINN

An AFINN can divide input–output data space and provide appropriate rules automatically. Fig. 1 shows the structure of AFINN. It consists of two layers. One is the input–output (I/O) layer and another is the rule-layer. The I/O layer consists of the input-part and the output-part. Each node in the rule-layer represents one fuzzy rule. Weights from the input-part to the rule-layer and those from the rule-layer to the output-part are fully connected and they store fuzzy if-then rules. Membership functions as premise part are expressed in the weights. Each weight from the rule-layer to the output-part corresponds to the estimated value of each rule. In short, the weights from the input-part to the rule-layer indicate if-parts of fuzzy if-then rules and those from the rule-layer to the output-part indicate then-parts. The shapes of membership functions are adjusted automatically in the learning phase.

2.2. Behavior of AFINN

Suppose that the number of neurons in the input-part, which is equal to the dimension of the input data, is N_1, the number of rules is N_2, and the number of neurons in the output-part, which is equal to the dimension of the output data, is N_3. The input data to the AFINN is expressed as follows:

$$\boldsymbol{U} = (u_1, u_2, \ldots, u_i, \ldots, u_{N_1}). \tag{1}$$

The subscripts i, j, and k refer to the nodes in the input-part, those in the rule-layer, and those in the output-part, respectively. Fig. 1(b) shows an example of a membership function. The bell-shaped membership function represents the if-part of fuzzy rule, which is placed between the ith input node and the jth node in the rule-layer. The membership function is expressed as

$$\mu_{ij} = \exp\left(-\frac{(u_i - w_{ij})^2}{\sigma_{ij}^2}\right), \quad i = (1, 2, \ldots, N_1), \quad j = (1, 2, \ldots, N_2), \tag{2}$$

where μ_{ij} is the membership value, w_{ij} is the center value of the membership function and σ_{ij} indicates the width adjusted in the parameter estimation phase explained in Section 3.4.

In the rule-layer, many conventional fuzzy systems calculate the degree of the rule by selecting minimum membership value or multiplying them as follows:

$$\rho_j = \min_i \mu_{ij}, \tag{3}$$

$$\rho_j = \prod_i^{N_1} \mu_{ij}. \tag{4}$$

These calculations, however, often tend to make ρ extremely small and sometimes they cause underflow when the dimension of the task is large. In such a situation, the learning and inference cannot be proceeded correctly. In order to solve such a problem, AFINN calculates the degree of the jth rule ρ_j as

$$\rho_j = \prod_i^{N_1} \mu_{ij}^{1/N_{adj}}. \tag{5}$$

Here, N_{adj} is the compensated factor relating to the input dimension. We discuss the efficiency of N_{adj} for high-dimensional data in Section 4.2.

Then, the inference result of the kth node in the output-part is calculated by the following equation:

$$\hat{y_k} = \frac{\sum_j^{N_2} (w_{jk}\rho_j)}{\sum_j^{N_2} \rho_j}, \quad k = (1, 2, \ldots, N_3), \tag{6}$$

where w_{jk} is the weight between the jth node in the rule-layer and the kth node in the output-part. The w_{jk} corresponds to the estimated value of the jth rule for the kth node in the output-part. The logical form of the fuzzy inference if-then rules is given such as

If u_1 is $\tilde{w}_{1j}$, and $\ldots, u_i$ is $\tilde{w}_{ij}, \ldots, u_{N_1}$ is $\tilde{w}_{N_1 j}$
then y_k is w_{jk},

where $\tilde{w}_{ij}$ means the value near w_{ij}. It should be noted here that it depends on the value of σ_{ij}.

3. Model construction of AFINN

AFINN has four phases to achieve appropriate input selection, rule creation and parameter adjustment. In this section we explain each of them.

3.1. Initial rule creation

The initial temporal rules are formed in this phase by adaptive self-organizing learning algorithm. This algorithm is developed based on that of Linkens [6] and is modified to the AFINN structure. It can construct variable structure in which the number of rules can be changed dynamically in response to incoming training data. The algorithm is expressed as follows:

Preparation: The lth input vector to the system $\boldsymbol{I}^l$ is defined as follows:

$$\boldsymbol{I}^l = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{Y} \end{bmatrix}. \tag{7}$$

Here, the vector $\boldsymbol{Y}$ is the desired data vector in the output-part of the I/O layer described as

$$\boldsymbol{Y} = (y_1^l, y_2^l, \ldots, y_k^l, \ldots, y_{N_3}^l)^{\mathrm{T}}. \tag{8}$$

The suffix l of $\boldsymbol{I}$ indicates the consecutive number of the learning vector ($1 \leqslant l \leqslant L$): the learning vector $\boldsymbol{I}$ consists of L sets of $(N_1 + N_3)$ dimensional vectors. They are normalized in [0,1] and inputted to the system.

The jth network weight vector $\boldsymbol{W}$ is defined to concatenate the weight vector from the ith input-part in the I/O layer to the jth node in the rule-layer, $\boldsymbol{w}_{ij} = (w_{1j}, w_{2j}, \ldots, w_{ij}, \ldots, w_{N_1 j})^{\mathrm{T}}$, and that from the jth node in the rule-layer to the kth output-part in the I/O layer, $\boldsymbol{w}_{ij} = (w_{j1}, w_{j2}, \ldots, w_{jk}, \ldots, w_{jN_3})^{\mathrm{T}}$. Note that $\boldsymbol{W}$ is $(N_1 + N_3)$-dimensional vector as well as $\boldsymbol{I}^l$.

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_{ij} \\ \boldsymbol{w}_{ij} \end{bmatrix}. \tag{9}$$

The number of current rule N_r is set to 0, the number of updated iterations for the jth rule defined as M_j is set to 0 $(\forall_j)$ and the consecutive number of the input vector l is set to 1.

Step 1: The first input vector $\boldsymbol{I}^1$ is used as the first rule $\boldsymbol{W}_1$.

$$\boldsymbol{W}_1 = \boldsymbol{I}^1. \tag{10}$$

N_r is set to 1 and $l = l + 1$.

Step 2: The winner rule j^* is selected from existing N_r rules where Euclidian distance between the lth input vector

$\boldsymbol{I}^l$ and the jth rule $\boldsymbol{W}$ is minimum.

$$\|\boldsymbol{W}-\boldsymbol{I}^l\| = \min_j^{N_r} \|\boldsymbol{W}-\boldsymbol{I}^l\|. \tag{11}$$

Step 3: If the Euclidian distance $\|\boldsymbol{W}-\boldsymbol{I}^l\|$ calculated in Step 2 is smaller than the threshold ξ_{SELF}, the weights of winner rule $\boldsymbol{W}$ is updated and $M_{j^*} = M_{j^*}+1$. Otherwise this lth input vector is registered as the new weight vector and $N_r = N_r + 1$.

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \varepsilon_{SELF}^{j^*}(\boldsymbol{I}^l - \boldsymbol{W}(t)) \quad \text{where } \|\boldsymbol{W}-\boldsymbol{I}^l\| < \xi_{SELF}, \tag{12}$$

$$\boldsymbol{W}_{+1} = \boldsymbol{I}^l \quad \text{elsewhere}. \tag{13}$$

Here, ε_{SELF}^{j} is the learning factor defined as follows:

$$\varepsilon_{SELF}^{j} = \varepsilon_{SELFinit} \cdot \left(\frac{L-M_j}{L}\right)^2, \tag{14}$$

where $\varepsilon_{SELFinit}$ is the initial learning constant.

Step 4: If $l = L$: This self-organizing learning phase has finished and initial fuzzy rules of AFINN have been created.

else: $l = l + 1$; continue steps 2 and 3 during $l \leqslant L$.

3.2. Input selection

It is necessary to select adequate input elements in order to construct suitable fuzzy model. Unnecessary inputs interfere with proper learning, inference and creation of efficient rules. AFINN refers to the procedure of Linkens et al. [6,7] that deals with multi-input–single-output system and extends this algorithm for multi-input–multi-output system. This procedure is composed of two steps. In the first step, importance of each input element for the outputs is estimated and elimination of unnecessary inputs is carried out. Then the correlations among remaining inputs are calculated. Based on these values, several groups in which correlations among each input element exceed the threshold each other are created. In each group, one element is selected as the representative input.

3.2.1. Calculation of importance of the inputs

First, the lth input importance value is calculated using network initial weights acquired in Section 3.1.

$$z_{ik}^l = \frac{\sum_j^{N_2} \mu_{ij} w_{jk}}{\sum_j^{N_2} \mu_{ij}}, \tag{15}$$

where μ_{ij} is the membership value calculated by Eq. (2).

Next, the integrated input importance value from the ith input to the kth output is calculated using L sets of z_{ik}^l.

$$F_{ik} = R_{ik}/R_k, \tag{16}$$

where

$$R_{ik} = \max_l(z_{ik}^l) - \min_l(z_{ik}^l), \tag{17}$$

$$R_k = \max_i R_{ik}. \tag{18}$$

Finally, the syntactic input importance value is calculated by accumulating F_{ik}.

$$F_i = \sum_k^{N_3} F_{ik}. \tag{19}$$

The ith input element satisfying the following equation is regarded as unnecessary and eliminated:

$$F_i < \max_i F_i \times \xi_{input}. \tag{20}$$

Here, ξ_{input} is the threshold.

3.2.2. Calculation of the input correlation

Further input selection using the correlation of each input is mentioned here. Suppose a pair of remaining input u_a and u_b $(1 \leqslant a, b \leqslant N_1)$, the correlation between them $\varphi(a,b)$ is calculated as

$$\varphi(a,b) = \frac{1}{L}\frac{\sum_l^L (u_a^l - \bar{u_a})(u_b^l - \bar{u_b})}{\sigma_{u_a}\sigma_{u_b}}, \tag{21}$$

where

$$\bar{u_i} = \frac{1}{L}\sum_l^L u_i^l,$$

$$\sigma_{u_i} = \sqrt{\bar{u_i^2} - (\bar{u_i})^2}.$$

The correlation is calculated for every pair in remaining input elements. If there are several input elements whose correlation exceeds the threshold ξ_{corr} each other, they are regarded as a group. In each group, the ith input element having largest F_i is used as the representative.

3.3. Model identification

After the efficient input elements are selected, self-organizing learning is carried out again to determine the structure of AFINN such as the number of rules and initial weights. Although many clustering techniques have been explored regarding to the fuzzy modeling, above mentioned adaptive self-organizing learning explained in Section 3.1 is applied to the proposed AFINN. The reason is that the number of optimal rules is in most cases unknown in advance and this method can create appropriate rules dynamically according to the input. In addition this method is easier and faster than Kohonen's algorithm [8].

3.4. Parameter estimation

In this parameter estimation phase, the mean-square error between outputs of the network and the desired signals is reduced to adjust the parameters such as the shape of the membership functions by LMS algorithm.

The goal of this phase is to minimize the following error function E. Here, we regard minimizing E as minimizing error function E_l, which indicates sum of errors for each learning pattern l.

$$E_l = \frac{1}{2} \sum_k^{N_3} (y_k - \hat{y}_k)^2, \tag{22}$$

$$E = \sum_l^{L} E_l, \tag{23}$$

where, y_k is the desired output at the kth node in the output-part and $\hat{y}_k$ is the output inferred by AFINN. Note that the suffix l for y_k and $\hat{y}_k$ are omitted here.

Suppose that x is a parameter to be adjusted, the LMS learning algorithm can be expressed as

$$x(t+1) = x(t) - \varepsilon_{LMS} \frac{\partial E}{\partial x}, \tag{24}$$

where ε_{LMS} is the learning factor.

According to the LMS learning principle, the estimation value of the jth rule node is updated as

$$w_{jk}(t+1) = w_{jk}(t) + \varepsilon_{LMS}(y_k - \hat{y}_k) \frac{\rho_j}{\sum_n^{N_2} \rho_n}. \tag{25}$$

The center value and the width of the jth rule are also updated as

$$w_{ij}(t+1) = w_{ij}(t) + \varepsilon_{LMS} \sum_k^{N_3} (y_k - \hat{y}_k) \times \left(\frac{w_{jk} \sum_n^{N_2} \rho_n - \sum_n^{N_2} (w_{nk}\rho_n)}{(\sum_n^{N_2} \rho_n)^2} \right) \times \frac{1}{N_{adj}} \rho_j \frac{2(u_i - w_{ij})}{\sigma_{ij}^2} \tag{26}$$

and

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \varepsilon_{LMS} \sum_k^{N_3} (y_k - \hat{y}_k) \times \left(\frac{w_{jk} \sum_n^{N_2} \rho_n - \sum_n^{N_2} (w_{nk}\rho_n)}{(\sum_n^{N_2} \rho_n)^2} \right) \times \frac{1}{N_{adj}} \rho_j \frac{2(u_i - w_{ij})^2}{\sigma_{ij}^3}. \tag{27}$$

These gradient decent based algorithms sometimes make the width of membership function negative during the learning. One solution for this phenomenon is introduction of the minimum threshold for the width of the membership function. However, this sometimes obstructs the learning of the systems in which Eq. (3) or similar inference algorithm is used to calculate the degree of the rule. This algorithm updates only weights associated with the selected rule j, so when the jth rule meets this threshold, the learning is seriously affected. The systems whose algorithm based on Eqs. (4) and (5) hardly meet this problem, however, the former tend to make ρ extremely small and often fail the inference when the dimension of the task is large as mentioned before. AFINN uses Eq. (5) to carry out learning and inference properly and introduces the limitation $\xi_{min\sigma}$ to keep the created rules meaningful.

4. Experimental results

We used the following three examples to verify the effectiveness of the proposed AFINN.

4.1. Formula approximation and model construction

We tested a non-linear 5-input–3-output system including one redundant input in order to check the model construction ability of AFINN.

$$y_1 = u_1^3 + 3u_2 - 4 \sin u_3,$$
$$y_2 = \cos^2 u_1 - \tan^3 u_2 + u_4,$$
$$y_3 = \sqrt{u_1 + u_2 + u_4}. \tag{28}$$

The 5th input u_5 is used as a redundant random input. Thousand sets of input vectors and the corresponding outputs are normalized in [0-1] are inputted to the AFINN. System parameters of this test are shown in Table 1 and the results of model construction under this condition are shown in Table 2. Note that F_i in Table 2 indicates the importance of the ith input element calculated in Eq. (19). From this table, we can see that the indispensable four elements are correctly selected and that redundant input is not selected in all cases.

Table 1
Parameters used in the formula approximation

N_1 (initial)	5
N_3	3
N_{adj}	$N_1/4$
L (No. of vectors)	1000
ξ_{SELF}	5–20%
$\varepsilon_{SELFinit}$	0.5
ξ_{input}	0.2
$\xi_{corr.}$	0.8
ε_{LMS}	0.001
σ_{init}	0.1
$\xi_{\min_\sigma}$	0.01
No. of LMS learning	5000

Table 2
Results of the model construction

ξ_{SELF} (%)	5.0	7.0	10.0	12.0	15.0	20.0
Initial No. of rules	815	509	208	127	61	23
F_1	0.30	0.30	0.29	0.25	0.33	0.32
F_2	0.57	0.57	0.58	0.59	0.60	0.59
F_3	0.22	0.22	0.25	0.25	0.25	0.25
F_4	0.28	0.27	0.25	0.24	0.22	0.25
F_5	0.03	0.03	0.03	0.03	0.04	0.11
Selected input	u_1, u_2	u_1, u_2	u_1, u_2	u_1, u_2	u_1, u_2	u_1, u_2
Elements	u_3, u_4	u_3, u_4	u_3, u_4	u_3, u_4	u_3, u_4	u_3, u_4
No. of rules (N_2)	527	254	80	44	21	8

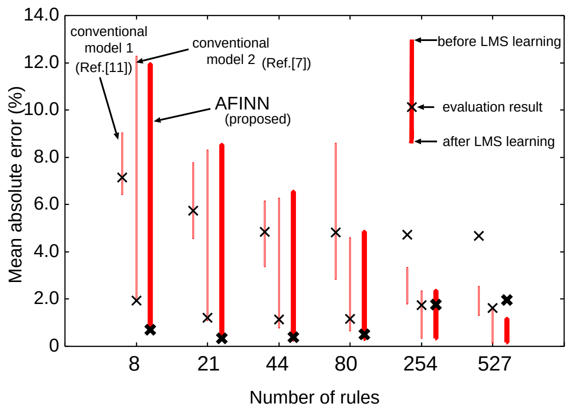

Fig. 2. Comparison of the error convergence and evaluation results.

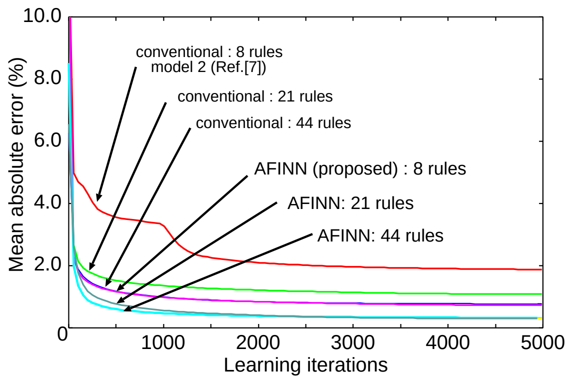

Fig. 3. Comparison of the error convergence curve on formula approximation.

Fig. 2 compares the results of parameter adjustment and the evaluation results for 300 unlearned data on the proposed AFINN and the conventional models. The conventional model 1 is FINN [11] and the conventional model 2 is similar to Linkens' system [7] which carries out input selection, rule creation and uses traditional min-max fuzzy inference algorithm shown by Eq. (3). Note that the conventional model 1 (FINN) cannot carry out input selection so that this result is affected by redundant input. When $\xi_{SELF} = 15.0\%$, AFINN created 21 rules and achieved 0.35% mean absolute error (MAE) on percentage for the evaluation data. On the other hand, conventional model 1 and 2 achieved 5.75% and 1.20% MAE, respectively. Fig. 3 compares the learning proceeding between the AFINN and the conventional model 2. Table 3 summarizes the evaluation results using several performance indexes, such as root mean square error (RMSE) and correlation. From these results, the proposed learning algorithm shows superior error convergence ability and evaluation results for unlearned data.

4.2. Speech recognition example

Second, we chose speech recognition example to examine the AFINN can handle the task whose input dimension is over hundreds correctly. We used ISOLET (isolated letter speech recognition) database from UCI machine learning repository [18]. Each record of ISOLET is composed of 617 elements and their corresponding spoken alphabet: this database can be used as 617-input 26-output classification task. This database consists of around 240 records for each alphabet, 6238 data in total. For the evaluation data set, the other 60 records are used for each alphabet, 1560 data in total also provided from UCI were used. Table 4 shows the parameters of this experiment.

Table 3
Performance comparison on formula approximation (training data/evaluation data)

	MAE	RMSE	Correlation
Conventional 1 [11]	4.35/5.25	0.31/0.33	0.963/0.952
Conventional 2 [7]	1.08/1.20	0.08/0.19	0.985/0.983
AFINN (proposed)	0.32/0.35	0.03/0.12	0.999/0.999

Under this experimental condition the 296 elements were selected from 617 inputs and 450 rules were created. Each MAE of before/after parameter adjustment and the results for the evaluation data were 3.47%, 0.49% and 1.00%,

Table 4
Parameters used in the speech recognition

N_1 (initial)	617
N_3	26
N_{adj}	$N_1/4$
L (No. of vectors)	6338
ξ_{SELF}	12%
$\varepsilon_{SELFinit}$	0.5
ξ_{input}	0.2
$\xi_{corr.}$	0.8
ε_{LMS}	0.001
σ_{init}	0.1
$\xi_{\min_\sigma}$	0.01
No. of LMS learning	100

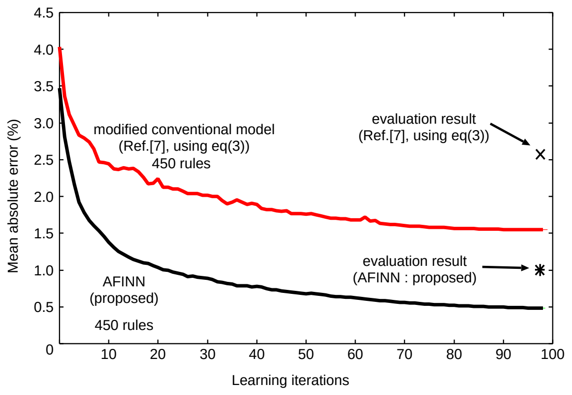

Fig. 4. Comparison of the learning and evaluation results on speech recognition.

respectively. The alphabetic classification accuracy for the evaluation data set was 93.53%. Even though this accuracy is a little worse than the best result by the conventional back propagation algorithm (95.9%) [19], the proposed system has several merits of fuzzy systems: linguistic rules extraction and their maintenance are possible.

Fig. 4 compares the error reduction and the evaluation results between the conventional model 2 and the proposed AFINN. The AFINN shows the superior results on error convergence and evaluation. It should be noted here that the system using traditional learning algorithm based on Eq. (4) caused underflow of the degree of rules and could not estimate the system parameters under such high input dimension. Therefore, in this experiment the conventional model 2 [7] uses Eq.(3) instead of Eq. (4) for inference algorithm.

Next, we compare the effectiveness of the compensated factor N_{adj}. Table 5 shows the residual error, evaluation error and correlation between given answer and the output on each N_{adj}. From this experiment, $\frac{1}{6}N_1 \leqslant N_{adj} \leqslant \frac{1}{3}N_1$ seemed to be reasonable and we selected $N_{adj} = \frac{1}{4}N_1$ for this factor. Introducing and selecting proper N_{adj} contributes to improve system availability and accuracy and

Table 5
The effectiveness of N_{adj} on speech recognition

N_{adj}	MAE	RMSE	Correlation
1 (Eq. (4): Ref. [7])	—/—	—/—	—/—
$\frac{1}{6}N_1$	0.40/0.79	0.08/0.09	0.969/0.937
$\frac{1}{4}N_1$ (proposed.)	0.38/0.75	0.08/0.09	0.970/0.939
$\frac{1}{3}N_1$	0.39/0.89	0.08/0.09	0.970/0.937
$\frac{1}{2}N_1$	0.68/0.94	0.09/0.10	0.953/0.928
Eq. (3): modified Ref. [7]	0.83/2.12	0.12/0.13	0.926/0.793

Table 6
Parameters used in the car evaluation

N_1 (initial)	6
N_3	4
N_{adj}	$N_1/4$
L (No. of vectors)	1382
ξ_{SELF}	20%
$\varepsilon_{SELFinit}$	0.5
ξ_{input}	0.2
$\xi_{corr.}$	0.8
ε_{LMS}	0.001
σ_{init}	0.1
$\xi_{\min_\sigma}$	0.01
No. of LMS learning	5000

we have confirmed similar results were obtained in other experiments.

4.3. *Car evaluation example*

As the last example, we chose car evaluation database also from UCI to confirm the validity of extracted rules. Each record of this database consists of six car characteristics and its evaluation result. The six input elements are buying price, maintenance cost, number of doors, number of passengers, luggage size and safety. Each input characteristics is classified from 3 to 5 grades. The corresponding 4 outputs are very good, good, acceptable and unacceptable. This database is composed of total 1728 data; 65, 69, 384 and 1210 data for each grade, respectively. We used 80% of them for training and used the remainder for the evaluation.

Table 6 shows the parameters of this experiment. When $\xi_{SELF} = 20.0\%$, all 6 inputs were selected and 10 rules are created. Table 7 shows the extracted all 10 rules. Under this condition, the AFINN achieved 3.19% MAE and 96.24% classification accuracy for the evaluation data set. The latest report using this database [20] showed almost 100% accuracy for this classification task. However, this method needs to divide the task into smaller ones manually and it just shows the classified results: it does not provide rules or other information.

Table 7
Extracted rules

Rule No.	Price	Maintenance cost	Number of doors	Number of passengers	Luggage size	Safety	Estimation
1	Very high	Med	4	4	Med	Med	Unacceptable
2	Very high	Med	4	$5\leqslant$	Small	Low	Unacceptable
3	High	Med	$5\leqslant$	4	Med	High	Acceptable
4	Low	Low	3	2	Med	High	Unacceptable
5	Low	High	$5\leqslant$	4	Large	High	Acceptable
6	Low	Low	4	4	Small	Med	Acceptable
7	Low	Low	4	4	Large	Low	Unacceptable
8	Low	Med	4	$5\leqslant$	Large	High	Very good
9	Med	Low	4	$5\leqslant$	Med	Med	Good
10	Low	Med	2	$5\leqslant$	Small	High	Unacceptable

In this data set, the high-ranked cars whose grade "good" or "very good" tend to have the following features: low price, low maintenance cost, many doors and passengers, large luggage capacity and high safety. Reasonable rules were obtained by the proposed AFINN.

5. Conclusions

In this paper we have proposed adaptive fuzzy inference neural network (AFINN). It has many desirable features such as simple structure, adequate self-construction, appropriate parameter adjusting and rule creation ability.

AFINN carries out efficient input selection, rule creation and parameter estimation to create appropriate fuzzy model and solves fuzzy-neuro specific problems by modification of the fuzzy inference and learning algorithm. With the use of formula approximation, speech recognition and car evaluation examples, we confirmed AFINN could construct relevant fuzzy model, conduct accurate inference and create appropriate rules correctly. In addition, as AFINN alleviates shortcomings of the conventional fuzzy neural network models, it can handle over hundreds dimensional tasks correctly. On account of its simple and applicable structure, AFINN can be used as a basic component for many applications.

Acknowledgements

This work is partly supported by the grants-in-aid from Ministry of Education, Culture, Sports, Science and Technology (C-12650394).

References

[1] T. Takagi, M. Sugeno, Structure identification of systems and its application to modelling and control, IEEE Trans. Systems Man Cybern. 15 (1985) 116–132.

[2] C.-T. Lin, C. Lee, Neural-network-based fuzzy logic control and decision systems, IEEE Trans. Comput. (special issue on Artificial Neural Networks) 40 (1991) 1320–1336.

[3] A. Amano, T. Arisuka, On the use of neural networks and fuzzy logic in speech recognition, in: Proceedings of the International Joint Conference on Neural Networks,1989, pp. 301–305.

[4] Y. Lin, G.A. Cunningham, A new approach to fuzzy-neural system modeling, IEEE Trans. Fuzzy Systems 3 (1995) 190–197.

[5] C. Wong, C.C. Chen, A hybrid clustering and gradient descent approach for fuzzy modeling, IEEE Trans. Systems, Man Cybern. 29 (1999) 686–693.

[6] D.A. Linkens, M.-Y. Chen, Input selection and partition validation for fuzzy modeling using neural network, Fuzzy Sets Systems 107 (1999) 299–308.

[7] M.-Y. Chen, D.A. Linkens, A systematic neuro-fuzzy modeling framework with application to material property prediction, IEEE Trans. Systems, Man Cybern. 31(5) (2001) 781–790.

[8] T. Kohonen, The self-organizing map, Proc. IEEE 78(9) (1990) 1464–1480.

[9] Jyh-Shing, R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Systems, Man Cybern. 23(3) (1993) 665–685.

[10] Li-Xin Wang, Training of fuzzy logic systems using nearest neighborhood clustering, in: Proceedings of the Second IEEE International Conference on Fuzzy Systems, Vol. 1, 1993, pp. 93–100.

[11] T. Nishina, M. Hagiwara, Fuzzy inference neural network, Neurocomputing 14 (1997) 223–239.

[12] H. Iyatomi, M. Hagiwara, Knowledge extraction from scenery images and the recognition using fuzzy inference neural networks, Trans. IEICE (D-II) J82-D-II(4) (1999) 685–693 (in Japanese).

[13] H. Iyatomi, M. Hagiwara, Scenery image recognition and interpretation using fuzzy inference neural networks, Pattern Recognition 35(8) (2002) 1793–1806.

[14] S.L. Chiu, Selecting input variables for fuzzy models, J. Intell. Fuzzy Systems 4 (1996) 243–256.

[15] C.-F. Juang, C.-T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Systems 6 (1998) 12–32.

[16] M. Sugeno, T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, IEEE Trans. Fuzzy Systems 1 (1993) 7–31.
[17] N.R. Pal, J.C. Bedzek, On cluster validity for c-means model, IEEE Trans. Fuzzy Systems 3 (1995) 370–379.
[18] UCI Machine Larning Repository, http://www.ics.uci.edu/ mlearn/.
[19] M. Fanty, R. Cole, Advances in Neural Information Processing Systems, Vol. 3, San Mateo, CA, 1991
[20] B. Zupan, M. Bohanec, I. Bratko, J. Demsar, Machine learning by function decomposition, ICML-97, Nashville, TN, 1997.

About the Author — HITOSHI IYATOMI is a research associate of Hosei University. He was born in Tokyo, Japan, on 25 March 1976. He received his B.E. and M.E. degrees in Electrical Engineering in 1998 and 2000 and Ph.D. degree in Open and Environmental Systems in 2004 from Keio University, respectively. From 2000–2004 he was employed by Hewlett Packard Japan. Since 2002 he has been COE (center of excellence) researcher at Keio University. His research interests include image understanding, machine learning, and medical image analysis.

About the Author — HAGIWARA MASAFUMI is a Professor of Keio University. He was born in Yokohama, Japan, on 29 October 1959. He received the B.E., M.E., and Ph.D. degrees in Electrical Engineering in Keio University, Yokohama Japan, in 1982, 1984 and 1987, respectively. In 1987, he became a research associate of Keio University. Since 1995, he has been an Associate Professor. From 1991 to 1993 he was visiting scholar at Stanford University. He received the Niwa Memorial Award, Shinohara Memorial Young Engineer Award, IEEE Consumer Electronics Society Chester Sall Award and Ando Memorial Young Engineer Award in 1986, 1987, 1990 and 1994, respectively. His research interests include soft computing. Dr. Hagiwara is a member of the IEEE, IEICE, IEE of Japan, IPSJ (Information processing society of Japan) and JNNS.