Multi-FNN Identification Based on HCM Clustering and Evolutionary Fuzzy Granulation

Ho-Sung Park and Sung-Kwun Oh

Abstract: In this paper, we introduce a category of Multi-FNN (Fuzzy-Neural Networks) models, analyze the underlying architectures and propose a comprehensive identification framework. The proposed Multi-FNNs dwell on a concept of fuzzy rule-based FNNs based on HCM clustering and evolutionary fuzzy granulation, and exploit linear inference being treated as a generic inference mechanism. By this nature, this FNN model is geared toward capturing relationships between information granules known as fuzzy sets. The form of the information granules themselves (in particular their distribution and a type of membership function) becomes an important design feature of the FNN model contributing to its structural as well as parametric optimization. The identification environment uses clustering techniques (Hard C – Means, HCM) and exploits genetic optimization as a vehicle of global optimization. The global optimization is augmented by more refined gradient-based learning mechanisms such as standard back-propagation. The HCM algorithm, whose role is to carry out preprocessing of the process data for system modeling, is utilized to determine the structure of Multi-FNNs. The detailed parameters of the Multi-FNN (such as apexes of membership functions, learning rates and momentum coefficients) are adjusted using genetic algorithms. An aggregate performance index with a weighting factor is proposed in order to achieve a sound balance between approximation and generalization (predictive) abilities of the model. To evaluate the performance of the proposed model, two numeric data sets are experimented with. One is the numerical data coming from a description of a certain nonlinear function and the other is NOx emission process data from a gas turbine power plant.

Keywords: Multi-FNN (Fuzzy Neural Networks), information granules, evolutionary fuzzy granulation, linear fuzzy inference, HCM clustering, genetic algorithms (GAs), design methodology.

1. INTRODUCTION

The design of mathematical models is inherently associated with a panoply of complex and uncertain issues. Recently, much research has been accomplished by developing a broad range of fuzzy neural networks (FNNs) – systems that synergistically combine the capabilities of fuzzy sets of handling uncertain (granular) information and essential learning features of neural networks.

Fuzzy set theory has been introduced [1] to model uncertain and/or ambiguous characteristics present in any experimental data. Since its inception, the research of fuzzy logic has been a focal point of various endeavors and demonstrated many fruitful results both in theory and application.

In early approaches, the generation of fuzzy rules and the ensuing adjustments (optimization) of their membership functions were done by *trial and error* on a basis of available operator's experience. Subsequently, designers find it difficult to develop adequate fuzzy rules and membership functions to reflect the essence of the data. This became even more profound when dealing with multidimensional data. Moreover, some useful information easily gets lost or ignored when human operators articulate their experience in the form of linguistic rules. A collection of manually developed fuzzy rules usually proves to be suboptimal. Consequently, there has arisen a genuine need for a sound optimization environment to construct and/or adjust a collection of linguistic rules. While there has been an impressive array of neuro-fuzzy approaches, comprehensive solutions are still to be developed. Interestingly, in this synergistic arrangement of fuzzy sets and neural networks, they tend to

Manuscript received April 23, 2002; accepted March 24, 2003.This study was supported by the Korea Energy Management Corporation (2001).

Ho-Sung Park is with the Department of Electrical Electronic & Information Engineering, Wonkwang University, Korea. (e-mail:neuron@wonkwang.ac.kr).

Sung Kwun Oh is with the Department of Electrical Electronic & Information Engineering, Wonkwang University, Korea. (e-mail:ohsk@wonkwang.ac.kr).

compensate for disadvantages of these two technologies when being used in the context of fuzzy rule-based models. The essential advantage of neural networks lies in their adaptive nature and mechanisms of learning from historical data. In the context of rules, the learning concerns the parameters of the membership functions.

Takagi and Hayashi [2] proposed a fuzzy inference method driven by neural networks. Horikawa [3] discussed a neuro-fuzzy topology in which optimization is based on gradient-based update mechanisms. Imasaki [4] introduced structured neural networks whose fuzzy rules consist of premise network, inference network, and consequence network. Nomura [5] proposed an auto-tuning method of fuzzy inference based on the delta rule. The problems reported there concerned the number of membership functions that fluctuated throughout the learning. Furthermore, the numbers of fuzzy rules increase with the addition of extra variables.

In this paper, we use a generic FNN model based on linear inference method as a fuzzy inference method. The basic FNN combines fuzzy "If-then" rules with neural networks that are learned (optimized) by means of the standard back-propagation algorithm. The structure of the network is constructed by partitioning fuzzy input-output space for each individual input variable. While conceptually simple, this approach exhibits a certain drawback: eventual relationships existing between the variables cannot be captured in this manner and reflected in the form of the ensuing fuzzy sets. To deal with shortcomings, we propose an idea of Multi-FNNs. First, a HCM clustering algorithm is used that carries out the input-output data preprocessing of all input variables viewed *en block* and develops a family of submodels that cope with homogeneous subsets of experimental data. Next, we use GAs [6-8] to support an overall optimization of the network. We introduce an aggregate objective function [9] that takes into account both training data and testing data. This index aims at achieving a sound balance between approximation and prediction capabilities of the proposed model.

At the experimental end, the proposed model is applied to numerical data of some three-variable nonlinear function [3, 10-12] and NOx emission process data of a gas turbine power plant [13].

2. MULTI-FNN BASED ON INFORMATION GRANULATION

In this section, we elaborate on the architecture and design process of Multi-FNNs. The architecture of the Multi-FNNs is based on a compressive and efficient framework of information granules with the aid of the HCM clustering method, and comes with fuzzy granulation formed through the space partition of the input variables of each single-FNN.

2.1. Linear fuzzy inference-based FNN

Here we discuss a type of "if-then" rules along with its development mechanisms of which the conclusion part comes with a linear fuzzy inference.

In this sense, the conclusion is expressed in the form of a linear relationship between inputs and an output variable. The basic model of the proposed Multi-FNN comes in the form shown in Fig. 1.

The fuzzy sets formed in the individual spaces (variables) form a preprocessing block of the FNN. The improved speed of learning is attributed to the usage of this interface.

The notation used in Figure 1 requires some clarification. The "boxes" and "circles" denote units of the FNN while "N" identifies a normalization procedure applied to the membership grades of the input variable x_i. The output $f_i(x_i)$ of the "Σ" neuron is described by the nonlinear function f_i. (We do not restrict ourselves to standard sigmoid functions as being commonly encountered in conventional neural networks).

The output of the FNN $\hat{y}$ is governed by the following expression,

$$\hat{y} = f_1(x_1) + f_2(x_2) + \cdots f_m(x_m) = \sum_{i=1}^{m} f_i(x_i). \quad (1)$$

with m being the number of the input variables (viz. the number of the output f_i's of the "Σ" neurons in the network). As previously mentioned, FNN is implied by the introduced fuzzy partition of each input variable. In this sense, we can regard each f_i given by (1) as the following mapping (rule),

$$R^j : IF \ x_i \ is \ A_{ij} \ then \ y_{ij} = w0_{ij} + x_i wa_{ij}. \quad (2)$$

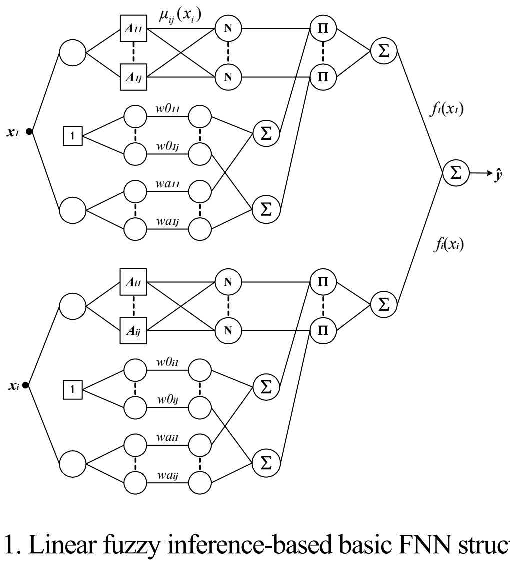


Fig. 1. Linear fuzzy inference-based basic FNN structure.

To be more specific, R^j is the *j-th* fuzzy rule while A_{ij} denotes a fuzzy variable of the premise of the corresponding fuzzy rule and represents membership function μ_{ij} as shown in Fig. 2. $w0_{ij}$ and wa_{ij} are constants of the consequence of the corresponding fuzzy rule. They express a connection (weight) existing between the neurons as we have already visualized in Fig. 1.

Furthermore we confine ourselves to triangular membership functions and make their membership grades sum up to 1 (so as to lead to a fuzzy partition of the variable). Each membership function in the premise part of the fuzzy rule is assigned to be complementary with neighboring ones in the form being shown in Fig. 2.

The numeric mapping from x_i to $f_i(x_i)$ is determined by fuzzy inferences and a standard defuzzification. The inference result coming from (2) follows a standard center of gravity aggregation.

$$f_i(x_i) = \sum_{j=1}^{n} \mu_{ij}(x_i)\cdot(w0_{ij} + x_i wa_{ij}) \Big/ \sum_{j=1}^{n} \mu_{ij}(x_i)\,. \quad (3)$$

In light of the properties of the fuzzy partition, we note that the input signal x_i activates only two neighboring membership functions labeled here by k and k+1 referred to in Fig. 2. Subsequently, (3) can be expressed as,

$$f_i(x_i) = \frac{\mu_{ik}(x_i)(w0_{ik} + x_i wa_{ik}) + \mu_{ik+1}(x_i)(w0_{ik+1} + x_i wa_{ik+1})}{\mu_{ik}(x_i) + \mu_{ik+1}(x_i)}. \quad (4)$$

The sum of the grades of these two neighboring membership functions labeled by k and k+1 is always equal to 1, that is, $\mu_{ik}(x_i) + \mu_{ik+1}(x_i) = 1$. Then (4) reduces to the form.

$$\begin{aligned} f_i(x_i) &= \mu_{ik}(x_i)\cdot(\mu 0_{ik} + x_i wa_{ik}) \\ &\quad + \mu_{ik+1}(x_i)\cdot(\mu 0_{ik+1} + x_i wa_{ik+1}) \end{aligned} \quad (5)$$

The learning of the FNN is realized by adjusting the connections of the neurons; the modifications of their values, $w0_{ij}$ and wa_{ij}, are accomplished through standard Back-Propagation (BP) algorithms. In this study, we use two measures (performance indexes).

- The use of the Euclidean error as a performance measure,

$$E_p = (y_p - \hat{y}_p)^2 \quad (6)$$

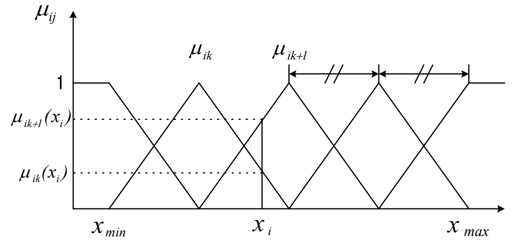


Fig. 2. MF definition before tuning (genetic optimization).

where, E_p is an error for the *p-th* data, y_p is the *p-th* target output data and $\hat{y}_p$ stands for the *p-th* actual output of the model for this specific data point. For m input-output data pairs, an overall (global) performance index comes as a sum of the errors.

$$E = \frac{1}{m}\sum_{p=1}^{m}(y_p - \hat{y}_p)^2. \quad (7)$$

- An average percentage-based error (APE) that assumes the form,

$$E = \frac{1}{m}\sum_{p=1}^{m}\frac{|y_p - \hat{y}_p|}{y_p}\times 100(\%). \quad (8)$$

As far as learning is concerned, the connections (here, wa) change in a standard fashion,

$$wa(new) = wa(old) + \Delta wa \quad (9)$$

where, the updated formula follows the gradient descent method, namely

$$\Delta wa_{ij} = \eta_a\left(-\frac{\partial E_p}{\partial wa_{ij}}\right) \quad (10)$$

with η_a being a positive learning rate.
Moreover we have,

$$-\frac{\partial E_p}{\partial wa_{ij}} = -\frac{\partial E_p}{\partial \hat{y}_p}\cdot\frac{\partial \hat{y}_p}{\partial f_i(x_i)}\cdot\frac{\partial f_i(x_i)}{\partial wa_{ij}}. \quad (11)$$

Combining (6) and (7), we derive,

$$-\frac{\partial E_p}{\partial \hat{y}_p} = -\frac{\partial}{\partial \hat{y}_p}(y_p - \hat{y}_p)^2 = 2(y_p - \hat{y}_p),$$

$$\frac{\partial \hat{y}_p}{\partial f_i(x_i)} = 1, \quad \frac{\partial C y_{ij}}{\partial wa_{ij}} = \mu_{ij}(x_i). \quad (12)$$

Finally, we obtain,

$$\Delta wa_{ij} = 2\cdot\eta_a\cdot(y_p - \hat{y}_p)\cdot\mu_{ij}(x_i) \quad (13)$$

Quite commonly to accelerate convergence, a momentum term is added to the learning expression. The momentum itself is defined in the form,

$$m(t) = wa_{ij}(t) - wa_{ij}(t-1) \quad (14)$$

Finally, the complete updated formula combining the already discussed components reads as,

$$\begin{aligned} wa_{ij} &= 2\cdot\eta_a\cdot(y_p - \hat{y}_p)\cdot\mu_{ij}(x_i)\cdot x_i \\ &\quad + \alpha_a(wa_{ij}(t) - wa_{ij}(t-1)) \end{aligned} \quad (15)$$

Here, the momentum coefficient, α_a is confined to the unit interval.

2.2. Multi-FNN structure

Conventional FNNs use HCM (Hard C-Means) clustering to determine values of some initial parameters of membership function through partition of the input space of systems [14]. In this study, we develop individual FNNs on the basis of some clusters of data being constructed through HCM clustering. The number of clusters corresponds to the number of the models (FNN structures), see Fig. 3. A bank of FNNs is used in parallel to build an overall model. Note that the data belonging to different clusters support the construction of the individual models.

2.3. HCM clustering method

It is worth emphasizing that the HCM clustering method has been used extensively not only to organize and categorize data, but it becomes useful in data compression and model identification. For the sake of completeness of the entire discussion, let us briefly recall the essence of the HCM algorithm.

Suppose that we are given a set of data X={$\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$}, where $\mathbf{x}_k$ =[x_{k1}, …, x_{km}], n is the number of data and m is the number of variables. Let P(X) be the power set of X, that is, the set of all the subsets of X. A hard c-partition of X is the family $\{A_i \in P(X): 1 \le i \le c\}$ such that $\bigcup_{i=1}^{c} A_i = X$ and $A_i \cap A_j = \phi$ for $1 \le i \ne j \le c$. Each A_i is viewed as a cluster, so {A_1, …, A_c} partitions X into c clusters. The hard c-partition can be reformulated through the characteristic (membership) function of the element $\mathbf{x}_k$ in A_i. Specifically, define

$$u_{ik} = \begin{cases} 1, & \mathbf{x}_k \in A_i \\ 0, & \mathbf{x}_k \notin A_i \end{cases}$$

where, $\mathbf{x}_k \in X$, $A_i \in P(X)$ and i=1,2,⋯,n. Clearly, u_{ik}=1 means that $\boldsymbol{x}_k$ belongs to cluster A_i. Given the value of u_{ik}, we can uniquely determine a hard c-partition of X, and vice versa. The elements of the partition matrix u_{ik} satisfy the following three conditions:

$$u_{ik} \in \{0,1\}, \quad 1 \le i \le c, \quad 1 \le k \le n \tag{16}$$

$$\sum_{i=1}^{c} u_{ik} = 1, \quad \forall \mathrm{k} \in \{1,2,\cdots,\mathrm{n}\} \tag{17}$$

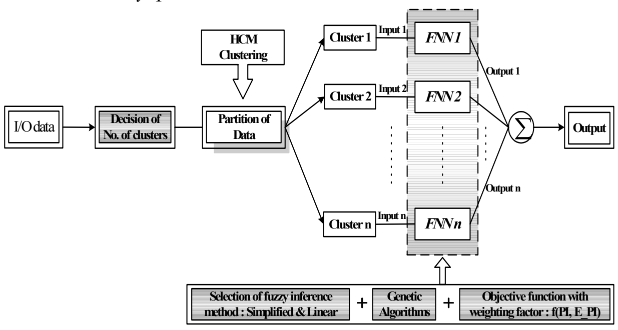


Fig. 3. Multi-FNN structure and ensuing development environment.

$$0 < \sum_{k=1}^{n} u_{ik} < n, \quad \forall \mathrm{i} \in \{1,2,\cdots,\mathrm{c}\} \tag{18}$$

At the interpretation end, (16) and (17) mean that each $\mathbf{x}_k \in X$ should belong to one and only one cluster. (18) requires that each cluster A_i must contain at least one and at most n-1 data point. By collecting u_{ik} with $1 \le i \le c$ and $1 \le k \le n$ into a c×n matrix **U**=[u_{ik}], we obtain the matrix representation for hard c-partition, defined as follows.

$$M_C = \left\{ \mathbf{U} \mid u_{ik} \in \{0,1\},\ \sum_{i=1}^{c} u_{ik} = 1,\ 0 < \sum_{k=1}^{n} u_{ik} < n \right\} \tag{19}$$

Step 1: Fix the number of clusters $c(2 \le c < n)$ and initialize the partition matrix $\mathbf{U}^{(0)} \in M_C$

Step 2: Calculate the center vectors $\mathbf{v}_i$ of each cluster:

$$\mathbf{v}_i^{(r)} = \{v_{i1}, v_{i2}, \cdots, v_{ij}, \cdots, v_{im}\} \tag{20}$$

$$v_{ij}^{(r)} = \sum_{k=1}^{n} u_{ik}^{(r)} \cdot x_{kj} \Bigg/ \sum_{k=1}^{n} u_{ik}^{(r)}$$

Where, [u_{ik}]= $\mathbf{U}^{(r)}$, i = 1, 2, …,c, j=1, 2, …,m.

Step 3: Update the partition matrix $\mathbf{U}^{(r)}$; these modifications are based on the standard Euclidean distance function between the data points and the prototypes,

$$d_{ik} = d(\mathbf{x}_k - \mathbf{v}_i) = \|\mathbf{x}_k - \mathbf{v}_i\| = \left[\sum_{j=1}^{m} (x_{kj} - v_{ij})^2 \right]^{1/2} \tag{21}$$

$$u_{ik}^{(r+1)} = \begin{cases} 1 & d_{ik}^{(r)} = \min\{d_{jk}^{(r)}\} \text{ for all } j \in c \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

Step 4: Check the termination criterion. If

$$\| \mathbf{U}^{(r+1)} - \mathbf{U}^{(r)} \| \le \varepsilon \text{ (tolerance level)} \tag{23}$$

Stop; otherwise set r=r+1 and return to Step 2

According to the procedure, training data is partitioned as several groups based on its characteristics.

We calculate the distance between the center vector of each partitioned training data group and testing data by (21), and then testing data are partitioned close to the center vector of each partitioned training data group. Here, $\mathbf{v}_i$ is the center value of the training data, but when we partition the training data, $\mathbf{x}_k$ represents the training data and when we partition the testing data, $\mathbf{x}_k$ represents the testing data. Partitioned data are then used towards the design of the individual FNNs.

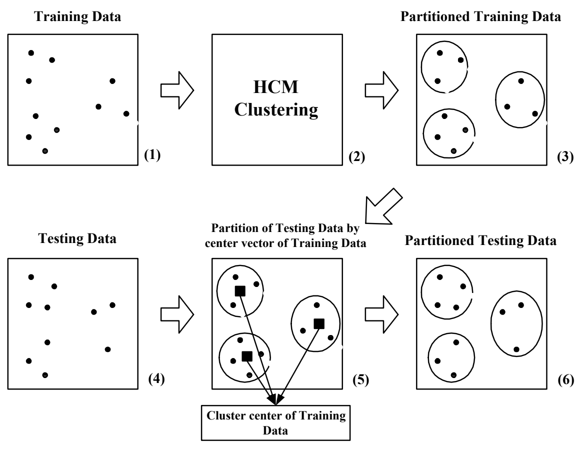

Training Data
HCM Clustering
Partitioned Training Data
(1)
(2)
(3)
Testing Data
Partition of Testing Data by center vector of Training Data
Partitioned Testing Data
(4)
(5)
(6)
Cluster center of Training Data

Fig. 4. Partition of training and testing data.

The partition of training and testing data is schematically visualized in Fig. 4. Partitioning of the training data is shown through phases (1) - (3), Fig. 4. The way in which the testing data are processed on the basis of the previously constructed cluster centers is displayed as phase (5). Finally, the testing data set is partitioned as visualized by (6) in Fig. 4.

3. OPTIMIZATION OF THE FNN MODEL

The task of optimizing any complex model involves two main problems. Firstly, a class of some optimization algorithms must be chosen so that it is applicable to the requirements implied by the problem at hand. Secondly, various parameters of the optimization algorithm must be tuned in order to achieve the best performance of the algorithm.

Genetic algorithms (GAs) are optimization techniques based on the principles of natural evolution. In essence, they are search algorithms that use operations found in natural genetics to guide a comprehensive search over the parameter space. GAs have been theoretically and empirically proven to provide robust search capabilities in complex spaces offering a valid approach to problems requiring efficient and effective searching.

To determine suitable values of the parameters for any given problem, GAs is developed.

3.1. Genetic algorithms

The need to handle optimization problems whose objective functions are complex and non-differentiable arises in many areas of system analysis and synthesis. While there are a number of analytic and numerical optimization techniques aimed at these tasks, there exists a wide range of problems that are out of reach by standard gradient-oriented techniques. Among objective functions that are highly challenging to these classical methods are those that are non-convex, multi-modal, and noisy [8].

Genetic algorithms [6-8] have proven to be useful in the optimization of such problems because of their ability to efficiently use historical information to obtain new solutions with enhanced performance and the global nature of search supported there. Genetic algorithms are also theoretically and empirically proven to support robust searches in complex search spaces. Moreover, they do not get trapped in local minima as opposed to gradient decent techniques being quite susceptible to this shortcoming. GAs are population-based optimization techniques.

The search of the solution space is completed with the aid of several genetic operators. There are three basic genetic operators used in any GA- supported search, reproduction, crossover and mutation. Reproduction is a process in which the mating pool for the next generation is chosen. Individual strings are copied into the mating pool according to their fitness function values. Crossover usually proceeds in two steps. First, members from the mating pool are mated at random. Second, each pair of strings undergoes crossover as follows: a position l along the string is selected uniformly at random from the interval [1, l-1], where l is the length of the string.

Two new strings are created by swapping all characters between the positions k and l. Mutation is a random alteration of the value of a string position. In a binary coding, mutation means changing a zero to a one or vice versa. Mutation occurs with small probability. Those operators, combined with the proper definition of the fitness function, constitute the main body of the genetic computation. A general flowchart of the genetic algorithm is shown in Fig. 5.

In this paper, for the optimization of the FNN model, GAs use the binary type serial method, roulette-wheel in the selection operator, one-point crossover in the crossover operator, and inversion in the mutation operator. Here, we use 100 generations, 60 populations, 10 bits per string, a crossover rate equal to 0.6, and mutation probability equal to 0.1. Fig. 7 shows how a string is composed in GAs. Where, variable x_1, x_2, …, and x_k denote input variables, learning

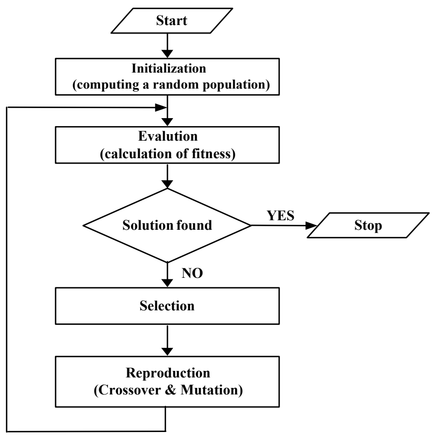

Start
Initialization (computing a random population)
Evalution (calculation of fitness)
Solution found
YES
Stop
NO
Selection
Reproduction (Crossover & Mutation)

Fig. 5. A general GA flowchart.

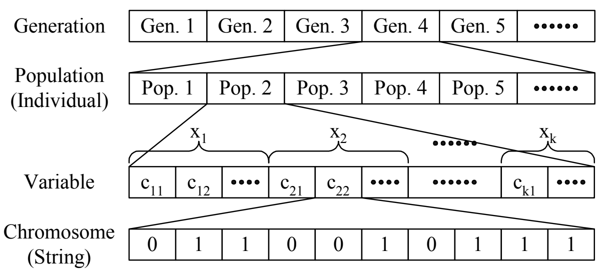


Fig. 6. Data structure of genetic algorithms for optimization of the FNN model.

rate, and momentum coefficient of the FNN model, c_{ij} denotes the vertical point of membership functions for each input variable, learning rate, and momentum coefficient.

3.2. The objective function with a weight factor

The objective function (performance index) is a basic instrument guiding the evolutionary search in the solution space [9]. The objective function includes both the training data and testing data and comes as a convex sum of two components,

$$f(PI,\ E\_PI) = \theta \times PI + (1 - \theta) \times E\_PI \quad (24)$$

PI and E_PI denote the performance index for the training data and testing data, respectively. Moreover θ is a weighting factor that allows us to strike a balance between the performance of the model for the training and testing data. Depending upon the values of the weighting factor, several specific cases of the objective function are worth distinguishing.

If $\theta = 1$ then the model is optimized based on the training data. No testing data is taken into consideration.

If $\theta = 0.5$ then both the training and testing data are taken into account. Moreover, it is assumed that they exhibit the same impact on the performance of the model.

The case $\theta = \alpha$ where $\alpha \in [0, 1]$ embraces both the cases stated above. The choice of α establishes a certain tradeoff between the approximation and generalization aspects of the FNN model.

The performance index used in the ensuing numerical experiments will be as Euclidean distance, see (7) and (8) while θ is regarded as an extra parameter.

4. SIMULATION AND DISCUSSION OF RESULTS

Once the identification methodology has been established, we proceed with intensive experimental studies. In this section, we provide three numerical examples to evaluate the advantages and the effectiveness of the proposal approach. These include the numerical data of three-input nonlinear function [3, 10-12] and the NOx emission process data of a gas turbine power plant [13]. The performance indexes (PI) used here are: (7) for NOx emission process data and (8) for the three-variable nonlinear function.

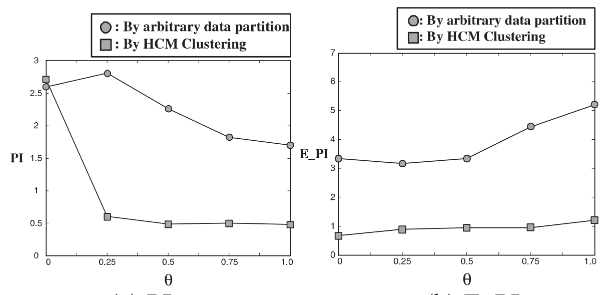
4.1. Three-input nonlinear function

In this experiment, we use the same numerical data as in the existing literature, cf. [3,10-12]. The nonlinear function to be modeled is expressed as

$$y = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2 . \quad (25)$$

We've considered 40 pairs [11] of the original input-output data. 20 out of the 40 pairs of input-output data are used as a learning set; the remaining portion serves as a testing set. The performance index is defined by (8). Here the number of membership functions for each input variable is set to three. As shown in Table 1, we can achieve enhanced performance of Multi-FNNs by the HCM data partition with c=2, in comparison to any arbitrary data partition.

Fig. 7 visualizes the final results (listed in Table 1) treated as a function of the weighting factor θ.



(a) PI (b) E_PI

Fig. 7. Performance index of Multi-FNN treated as a function of θ.

Table 1. Performance Index in the number of clusters, c=2.

θ	Arbitrary data partition					
	FNN 1		FNN 2		Final result	
	PI	E_PI	PI	E_PI	PI	E_PI
0.0	2.813	2.993	2.380	3.710	2.597	3.351
0.25	3.159	3.033	2.439	3.304	2.799	3.168
0.5	2.790	3.209	1.721	3.473	2.255	3.341
0.75	1.961	4.610	1.682	4.236	1.821	4.423
1.0	1.907	4.445	1.491	5.958	1.699	5.201
θ	Partitioned data by HCM clustering					
	FNN 1		FNN 2		Final result	
	PI	E_PI	PI	E_PI	PI	E_PI
0.0	1.057	1.143	4.356	0.197	2.706	0.670
0.25	0.491	1.200	0.711	0.574	0.601	0.887
0.5	0.304	1.274	0.666	0.615	0.485	0.945
0.75	0.381	1.270	0.614	0.636	0.497	0.953
1.0	0.320	1.311	0.633	1.097	0.477	1.204

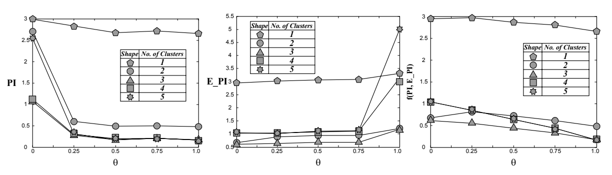


Fig. 8. Performance index of Multi-FNN treated as a function of the weighting factor for selected number of the clusters.

Table 2. Comparison of identification errors for selected fuzzy models.

MODEL			PI	E_PI
Shinichi's Model [3]	Type 1		0.84	1.22
	Type 2		0.73	1.28
Sugeno's Model [10]	Model I		1.5	2.1
	Model II		1.1	3.6
Linear model [11]			12.7	11.1
GMDH [12]			4.7	5.7
Single-FNN	Linear fuzzy inference	θ =0.5	2.670	3.063
		θ =1.0	2.652	3.309
Our model	Linear fuzzy inference	θ =0.5	0.174	0.689
		θ =0.75	0.210	0.679
		θ =1.0	0.168	1.182

Fig. 8 depicts the values of the performance index of Multi-FNNs treated as a function of the weighting factor and the number of the clusters.

The preferred results (that is PI=0.174, E_PI= 0.689) are reported when using three clusters and when setting the weighting factor (θ) to be equal to 0.5.

Table 2 contains a comparative analysis including several previous models. Sugeno's fuzzy models I and II are based on the linear inference method while Shinichi's models are fuzzy models obtained by using the back-propagation algorithm of fuzzy-neural networks. Compared with these models, the Multi-FNNs emerge as the ones with high accuracy and improved prediction capability.

4.2. NOx emission process of a gas turbine power plant

The NOx emission process is also modeled using the data from gas turbine power plants. Until now, almost all NOx emission processes are based on a "standard" mathematical model in order to obtain regulation data from the control process. However, such models do not develop the relationships between variables of the NOx emission process and parameters of its model in an effective manner. A NOx emission process of a GE gas turbine power plant located in Virginia, U. S. A., has been chosen in this modeling study.

The input variables include AT (Ambient Temperature at site), CS (Compressor Speed), LPTS (Low Pressure Turbine Speed), CDP (Compressor Discharge Pressure), and TET (Turbine Exhaust Temperature). The output variable is NOx [13]. The performance index is defined by (7).

Using NOx emission process data, the regression equation (model) reads as follows

$$\begin{aligned} y = &-163.77341 - 0.06709x_1 + 0.00322x_2 \\ &+ 0.00235x_3 + 0.26365x_4 + 0.20893x_5 \end{aligned} \tag{26}$$

This simple model comes with the value of PI=17.68 and E_PI=19.23. We will be using these results as a reference point when discussing FNN models.

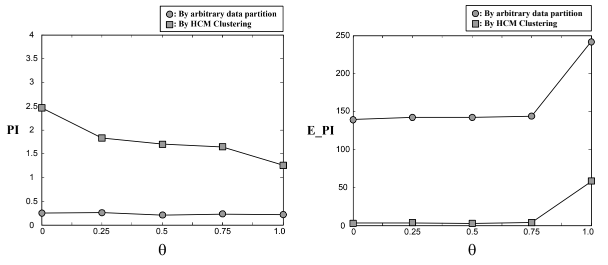
The comparison of the performance index of the HCM - partitioned data (c=2) with an arbitrary data partition is shown in Table 3. It becomes apparent that the clustering contributes to the improved performance of the model.

The performance of the model is also visualized in Fig. 9.

In the case of the NOx emission process of a gas turbine power plant, this dataset comes with four or more input variables and exhibits strong nonlinear

Table 3. Performance Index in case of data partitioning into two clusters, *c*=2.

θ	Arbitrary data partition					
	FNN 1		FNN 2		Final result	
	PI	E_PI	PI	E_PI	PI	E_PI
0.0	0.500	0.778	0	278.2	0.250	139.4
0.25	0.525	0.945	0.004	283.0	0.265	142.2
0.5	0.394	0.963	0.004	283.0	0.199	142.0
0.75	0.450	0.966	0.001	286.6	0.225	143.8
1.0	0.444	1.156	0	484.0	0.222	242.5
θ	Partitioned data by HCM clustering					
	FNN 1		FNN 2		Final result	
	PI	E_PI	PI	E_PI	PI	E_PI
0.0	0.907	1.297	4.570	5.507	2.457	3.013
0.25	0.777	1.597	3.270	5.086	1.832	3.019
0.5	0.860	1.416	2.843	4.927	1.699	2.847
0.75	0.702	1.618	2.924	6.791	1.642	3.727
1.0	0.746	1.951	0.162	139.5	1.256	58.04



(a) PI (b) E_PI

Fig. 9. Performance index of Multi-FNNs treated as a function of θ.

Table 4. Performance index vis-a-vis the changes in the number of MFs and the type of fuzzy inference method.

PI / No. of MFs per input	Tr_PI	Va_PI	Te_PI
2	14.509	13.700	4.605
3	8.763	10.924	3.613
6	3.810	5.782	2.457

Table 5. Performance Index according to the number of clusters.

θ	Single-FNN		Multi-FNN					
			Cluster 2		Cluster 3		Cluster 4	
	PI	E_PI	PI	E_PI	PI	E_PI	PI	E_PI
0.0	4.461	5.496	2.457	3.013	1.348	2.122	1.092	2.214
0.25	3.910	5.776	1.832	3.019	1.169	2.049	1.431	2.152
0.5	4.038	6.028	1.699	2.847	1.140	2.653	1.076	2.250
0.75	3.620	5.560	1.642	3.727	1.054	3.078	0.720	2.025
1.0	3.795	6.814	1.256	58.04	0.840	3.183	0.793	2.910

characteristics. To determine the performance of the fuzzy model, an overall dataset (260 pairs of I/O data) is split into three parts, namely a training dataset (100 pairs of I/O data), validation dataset (100 pairs of I/O data) and testing dataset (60 pairs of I/O data). Table 4 quantifies the values of the performance index when varying selected parameters of the model.

Here, Tr_PI, Va_PI and Te_PI denote the performance index for the training dataset, validation dataset and testing dataset, respectively. When the number of membership functions for each input variable in creases, we note enhanced performance of the model.

As shown in Table 4, in case of 6 MFs per input variable, the variation ratio (slope) of the performance index of the single-FNN model does not change radically. This led us to accept six fuzzy sets for each input variable. Equally as in other experiments, the NOx emission dataset is split into two parts, namely a training dataset and a testing dataset.

Fig. 10 depicts the values of the performance index of Multi-FNNs that are treated as a function of the weighting factor and the number of the clusters.

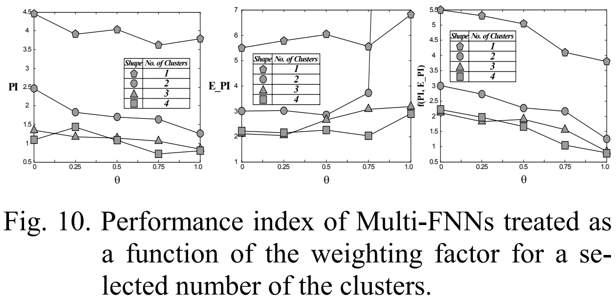


Fig. 10. Performance index of Multi-FNNs treated as a function of the weighting factor for a selected number of the clusters.

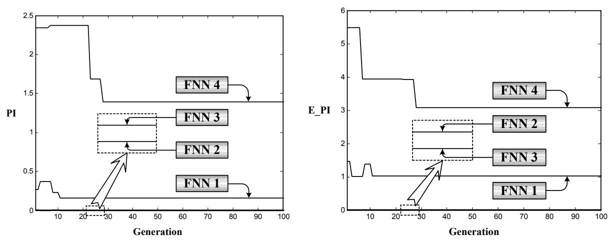


Fig. 11. Genetic optimization of the FNN model (θ =0.75).

Preferred results (that is PI=0.720, E_PI=2.025) are reported when using four clusters with the weighting factor (θ) equal to 0.75. Refer also to Fig. 10.

The results of the genetic optimization (four clusters) are shown in Fig. 11.

5. CONCLUSION

In this paper, we proposed Multi-FNNs as parallel-structures of fuzzy neural networks. The construction of Multi-FNNs dwells on the hybrid technologies combined with HCM clustering, linear fuzzy inference-based FNN, and genetic algorithms. Here, HCM clustering is used not only to analyze data that is characteristic of nonlinear and complex actual systems but also to construct a well-organized and effective model. Further, an efficient identification technique is presented that automatically extracts the optimal parameters of the FNN using the genetic algorithm and the weighting factor of an objective function. The underlying idea deals with the optimization of information granules by exploiting techniques of clustering and evolutionary computing.

The experimental studies clearly revealed that we could obtain superior performance (both approximation and generalization capabilities) for two commonly used experimental datasets.

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Inf. Control* 8, pp 338-353, 1965.

[2] H. Takagi and I. Hayashi, "NN-driven fuzzy reasoning," *Int. J. of Approximate Reasoning*, vol.5, no.3, pp. 191-212, 1991.

[3] S. Horikawa, T. Furuhashi, and Y. Uchigawa, "On fuzzy modeling using fuzzy neural networks with the back propagation algorithm," *IEEE trans. Neural Networks*, vol.3, no.5, pp. 801-806, 1992.

[4] N. Imasaki, J. Kiji, and T. Endo, "A fuzzy rule structured neural networks," *Journal of Japan Society for Fuzzy Theory and Systems*, vol.4, no.5, pp. 987-995, 1992 (in Japanese).

[5] H. Nomura and Wakami, "A self-tuning method of fuzzy control by descent methods," *4th IFSA'91*. pp.155-159, 1991.

[6] D. E. Golderg, *Genetic Algorithm in search, Op-*

timization & Machine Learning, Addison Wesley, 1989.

[7] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, 1996.

[8] D. Jong, K. A. *Are Genetic Algorithms Function Optimizers?*, Parallel Problem Solving from Nature 2, Manner, R. and Manderick, B. eds., North-Holland, Amsterdam.

[9] S. K. Oh and W. Pedrycz, "Identification of fuzzy systems by means of an auto-tuning algorithm and its application to nonlinear systems," *Fuzzy Sets and Syst.*, vol. 115, no. 2, pp. 205-230, 2000.

[10] G. Kang and M. Sugeno, "Fuzzy modeling," *Trans. SICE*, vol. 23, no. 6, pp. 106-108, 1987(in Japanese).

[11] M. Sugeno, *Fuzzy Control*. Tokyo, Japan:Nikkan Kogyo Shimbun-sha, 1988 (in Japanese).

[12] T. Kondo, "Revised GMDH algorithm estimating degree of the complete polynomial," *Tran. of the Society of Instrument and Control Engineers*, vol. 22, no. 9, pp. 928-934, 1986.

[13] G. Vachtsevanos, V. Ramani, and T. W. Hwang, *Prediction of Gas Turbine NOx Emissions using Polynomial Neural Network*, Technical Report, Georgia Institute of Technology, Atlanta, 1995.

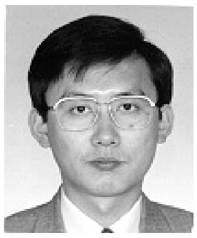
[14] S. K. Oh, K. C. Yoon, H. K. Kim, "The design of optimal fuzzy-neural networks structure by means of ga and an aggregate weighted performance index," *The Institute of Control, Automation and Systems Engineers(ICASE)*, vol. 6, no. 3, pp. 273-283, March 2000.

[15] B. J. Park, W. Pedrycz and S. K. Oh, "Identification of fuzzy models with the aid of evolutionary data granulation," *IEE Proc.-Control Theory and Applications,* vol. 148, Issue 05, pp. 406-418, September 2001.

[16] S. K. Oh, *Fuzzy Model & Control System by C-Programming*, Naeha press, 2002.

[17] S. K. Oh, *Computational Intelligence by Programming focused on Fuzzy, Neural Networks, and Genetic Algorithms*, Naeha press, 2002.

Ho-Sung Park received the B.S. and M.S. degrees in control and instrumentation engineering from Wonkwang University, Korea in 1999 and 2001, respectively. He is currently a Ph. D. student at the same institute. His research interests include fuzzy and hybrid systems, neurofuzzy models, genetic algorithms, and computational intelligence. He is a member of KIEE and ICASE.



Sung-Kwun Oh received the B.S., M.S., and Ph. D. degrees in electrical engineering from Yonsei University, Seoul, Korea, in 1981, 1983 and 1993, respectively. During 1983-1989, he worked as the Senior Researcher in the R&D Lab. of Lucky-Goldstar Industrial Systems Co., Ltd. He was a Postdoctoral fellow in the Department of Electrical and Computer Engineering at the University of Manitoba, Canada, from 1996 to 1997. He is currently an Associate Professor in the School of Electrical, Electronic and Information Engineering, Wonkwang University, Korea. His research interests include fuzzy systems, fuzzy-neural networks, automation systems, advanced Computational Intelligence, and intelligent control. He is a member of IEEE. He currently serves as an Associate Editor for the Korean Institute of Electrical Engineers (KIEE) and the Institute of Control, Automation & Systems Engineers (ICASE), Korea.