Fuzzy inference and rule extraction using a neural network

G. Castellano and A. M. Fanelli

Università degli Studi di Bari, Dipartimento di Informatica
Via E. Orabona, 4 - 70126 Bari - ITALY
E-Mail:[castellano,fanelli]@di.uniba.it

Abstract – *This paper proposes a neural network for building and optimizing fuzzy models. The network can be regarded both as an adaptive fuzzy inference system with the capability of learning fuzzy rules from data, and as a connectionist architecture provided with linguistic meaning. Fuzzy rules are extracted from training examples by a hybrid learning scheme comprised of two phases: rule generation phase from data using a modified competitive learning, and rule parameter tuning phase using gradient descent learning. This allows simultaneous definition of the structure and the parameters of the fuzzy rule base. After learning, the network encodes in its topology the essential design parameters of a fuzzy inference system. A well-known classification benchmark is used to illustrate applicability of the proposed neuro-fuzzy hybrid network.*

1 Introduction

Essentially, system modeling is the task of building models from a combination of a-priori knowledge and empirical data. When we are trying to model a complex system, usually the only available information is a collection of empirical data, which are inherently imprecise as obtained from the observation of the system behavior. One type of modeling with imprecise data is fuzzy modeling, whose objective is to extract a model in the form of fuzzy inference rules. Fuzzy modeling based on numerical data, which was first explored systematically by Takagi and Sugeno [1] has found numerous successful applications to complex system modeling. Generally speaking, there are two challenging design issues to be addressed in fuzzy modeling: structure and parameter identification of the fuzzy rule base. Structure identification amounts to determine the number of rules needed, i.e. finding how many rules are necessary and sufficient to properly model the available data, and the number of membership functions for input and output variables. Parameter identification concerns the definition of parameters of membership functions used to describe antecedent and consequent parts of fuzzy rules.

Considerable work has been done to integrate the excellent learning capability of neural networks with fuzzy inference systems, resulting in neuro-fuzzy modeling approaches that combine the benefits of these two powerful paradigms into a single capsule and provide a powerful framework to extract fuzzy rules from numerical data [2], [3], [4], [5], [6]. However, as pointed out in [7], many neuro-fuzzy design techniques address only the parameter identification issue, and discard the identification of the structure, which has to be specified in advance. To our knowledge, few results on simultaneous structure and parameter determination have been reported in literature so far [8], [9].

In this paper, a neural network is proposed that is able to perform both structure and parameter identification of the fuzzy rules, based on the available data. The network is trained by a hybrid learning scheme that combines a competitive learning phase with a gradient descent (supervised) learning phase to extract a set of fuzzy rules that adequately represent a given dataset. In the first phase, a modified competitive learning is employed to establish the structure and initial parameters of a fuzzy rule base. In contrast to most existing neuro-fuzzy approaches [8], [9], the proper number of fuzzy rules and membership functions (network structure size), together with initial values of rule parameters (network weights) are simultaneously determined in the first learning phase. In the second learning phase, parameters of the fuzzy rules are optimized via a gradient descent technique with an effort to improve the

performance of the derived fuzzy model. Once the learning is completed, the network architecture encodes the knowledge learned in the form of fuzzy rules and processes data following fuzzy reasoning principles.
Simulations on a well-known classification benchmark verify the effectiveness of the proposed network.

2 Architecture of the neural network

The architecture of the network proposed here realizes the inference mechanism of a zero-order Takagi-Sugeno fuzzy model, based on a collection of *H rules* of the form:

$$R_k : \text{IF } (x_1 \text{ is } A_1^k) \text{ AND...AND } (x_n \text{ is } A_n^k) \text{ THEN } (y_1 \textit{ is } v_{k1}) \textit{ AND...AND } (y_m \textit{ is } v_{km})$$

where R_k is the *k*th rule ($1 \le k \le H$), $\{x_i\}_{i=1...n}$ are the input variables, $\{y_j\}_{j=1...n}$ are the output variables, A_i^k are fuzzy sets defined on the input variables, and v_{kj} are fuzzy singletons defined on the output variables.

Fuzzy sets A_i^k are defined by bell-shaped (Gaussian) membership functions

$$\mu_{ik}(x_i) = \mathrm{e}^{-(x_i - w_{ik})^2 / \sigma_{ik}^2}$$

where w_{ik} and σ_{ik} are the center and the width of the Gaussian function, respectively.
By adopting singleton fuzzification, product rule inference and center average defuzzification, the inferred crisp output value of this fuzzy system for any input $\bar{x}^0 = (x_1^0, x_2^0, ..., x_n^0)$, is calculated as:

$$y_j^0 = \frac{\sum_{k=1}^{H} \mu_k\left(\bar{x}^0\right) \cdot v_{kj}}{\sum_{k=1}^{H} \mu_k\left(\bar{x}^0\right)} \qquad (1)$$

where $\mu_k(\bar{x}^0) = \prod_{i=1}^{n} \mu_{ik}(x_i^0)$ is the activation strength of the *k*th rule.
To realize the described fuzzy inference mechanism, we propose a neural network with three layers.

1. *Layer* L_1. Units in this layer receive the input values $(x_1, x_2, ..., x_n)$ and act as fuzzy sets representing the terms of the corresponding input variable. Nodes in this layer are arranged into *H* groups; each group representing the IF-part of a fuzzy rule. Each node $i_k \in L_1$ receives the input variable concerned, *i.e.* x_i and computes the membership value $\mu_{ik}(x_i)$ that specifies the degree to which the input value x_i belongs to the fuzzy set A_i^k. Hence, the output of node $i_k \in L_1$ is in the range [0, 1] and is computed by the following function:
$$f_{ik}^{(1)}(x_i) = \mathrm{e}^{-(x_i - w_{ik})^2 / \sigma_{ik}^2}$$

2. *Layer* L_2. The number of nodes in this layer is equal to the number of fuzzy rules. A node in this layer represents a fuzzy rule; for each node, there are *n* fixed links from the input term nodes representing the IF-part of the fuzzy rule. The *k*th node performs the AND operation for precondition matching of the *k*th rule by Larsen product operator; thus the output of this node is:
$$f_k^{(2)}(\bar{x}) = \prod_{i=1}^{n} f_{ik}^{(1)}(x_i)$$

3. *Layer* L_3. Nodes in this layer represent the output variables of the system. Each node *j* acts as a defuzzifier and computes the output values according to (1):

$$f_j^{(3)}(\bar{x}) = \frac{\sum_{k=1}^{H} f_k^{(2)}(\bar{x}) \cdot v_{kj}}{\sum_{k=1}^{H} f_k^{(2)}(\bar{x})}$$

This neuro-fuzzy network encodes a set of fuzzy rules in its topology, and processes information in a way that matches the fuzzy reasoning scheme adopted. The weights of the network correspond to the Gaussian membership functions parameters $\{w_{ik}\}, \{\sigma_{ik}\}$ and to the consequent singletons $\{v_{kj}\}$. In other words, each node $k \in L_2$ is associated with two premise weight vectors $\bar{w}_k = (w_{1k},...,w_{nk})$, $\bar{\sigma}_k = (\sigma_{1k},...,\sigma_{nk})$ and one consequent weight vector $\bar{v}_k = (v_{k1},...,v_{km})$. The architecture of this neuro-fuzzy inference network is depicted in figure 1, where nodes representing the premise part of a fuzzy rule are enclosed in a gray circle.

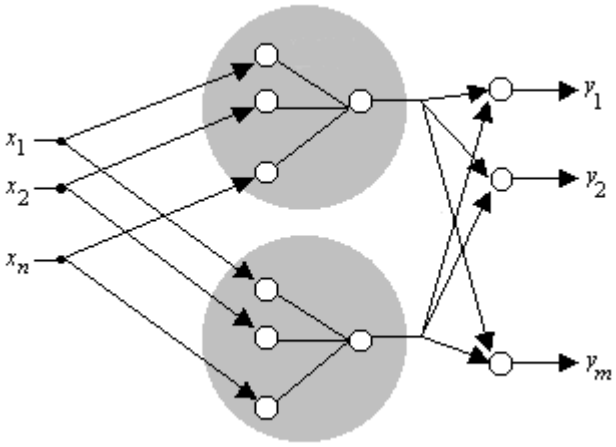

Figure 1. The neural fuzzy network.

3 Hybrid learning scheme

The learning scheme is mainly composed of two phases. In the first phase, the number of rule nodes (and hence the structure of the network) and initial rule parameters (weights) are determined using a competitive learning scheme; in the latter all parameters are adjusted using a supervised learning scheme. To initiate the learning scheme, a training set composed by input-output data samples, i.e. $S = \{(\bar{x}^p, \bar{y}^p), p = 1,...P\}$, and the desired or guessed coarse fuzzy partition, i.e. the initial number *K* of fuzzy rules, must be provided from the outside world. The schematic block diagram of the hybrid learning scheme is illustrated in figure 2.

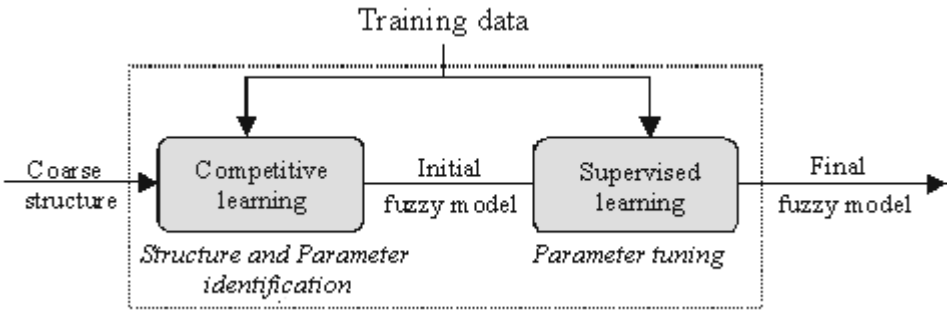

Figure 2. The hybrid learning scheme

3.1 Competitive learning phase

The first learning phase is aimed to find simultaneously the number of rules, and the antecedent and consequent parameters of each rule. This is achieved by clustering the input space through a modified competitive learning algorithm similar to [10], and deriving a fuzzy rule from each cluster. Unlike common clustering-based methods (e.g. *c*-means, fuzzy *c-means*, conventional competitive learning) which require the number of clusters, and hence the number of rules, to be appropriately pre-selected, the adopted modified competitive learning scheme performs clustering with the ability to adapt the number of clusters as the learning proceeds. Before this learning phase, an initial structure of the neuro-fuzzy network is first constructed based on the guessed number of rules given as a form of a-priori knowledge. Then, during learning, a rival penalized mechanism gradually drives the weight vectors (cluster centers) of extra nodes far away from the distribution of the data. In this way the appropriate number of rules for representing the input data is automatically selected.

Only the second layer of the network is involved in the competitive learning phase. When a n-dimensional vector $\bar{x}$ is presented, nodes in this layer compete and the node whose weight vector is closest to the vector $\bar{x}$ is chosen as winner. The closeness is measured in terms of Euclidean distance scaled up with the winning frequency of the node. The second closest node is marked as the *rival*. Then, the weight vector of the winning neuron is updated so as to become closer to the current input vector $\bar{x}$, while the weight vector of the rival is updated so as to move it away from the pattern. This mechanism tries to push the weight vector of the rival node far away from the cluster, towards which the weight vector of the winner is moving, thus implicitly making sure that each cluster is represented by only one weight vector. The use of this rival-penalized mechanism, that gradually drives the weight vectors of extra nodes far away from the distribution of the data, allows the appropriate number of nodes in the second layer, and hence the number of rules, to be automatically selected. The complete learning algorithm is described in Figure 3.

Starting with K nodes in the second layer, the modified competitive learning dynamically finds a set of H nodes ($H \le K$) whose weight vectors $\bar{w}_k$, $k = 1,...,H$ represent the centers of clusters in the input space. The components of each weight vector $\bar{w}_k$ represent the initial center values of the Gaussian membership functions in the premise part of the kth rule. To complete the initialization of premise parameters of fuzzy rules, the weights $\{\sigma_{ik}\}$ representing the widths of the membership functions must be derived. They are obtained using the *N-first-nearest-neighbor heuristic*:

$$\sigma_{ik} = \frac{\|\bar{w}_k - \bar{w}_h\|}{r} \qquad \text{for } i = 1,...,n$$

where $\bar{w}_h$ is the cluster center nearest to $\bar{w}_k$ and r is an *overlap* parameter ranging in $[1.0,2.0]$.

Finally, initial values of weights $\{v_{kj}\}$ representing rule consequent parameters are obtained using the cluster membership values $\mu_k(\bar{x}^p)$ and the target vectors $\bar{y}^p = (y_1^p \,....\, y_m^p)$, for all training patterns $p = 1,...,P$ as follows:

$$v_{kj} = \frac{\sum_{p=1}^{P} \mu_k\left(\bar{x}^p\right) \cdot y_j^p}{\sum_{p=1}^{P} \mu_k\left(\bar{x}^p\right)}$$

Hence, during the first learning phase, the structure as well as the initial weights of the neuro-fuzzy network are established. This corresponds to determine the number of fuzzy rules and the initial parameters of each rule. This initial fuzzy rule base is used as starting point for the subsequent supervised learning phase to obtain the final fuzzy rule base.

/ Competitive learning */*

1. *Input:* Randomly take an input vector $\bar{x}$ from a data set *S*, and compute:

$$d\left(\bar{x}, \bar{w}_k\right) = \frac{n_k}{\sum_{t=1}^{K} n_t} \left\|\bar{x} - \bar{w}_k\right\| \qquad \text{for } k = 1.....K$$

   n_k is the cumulative number of the winning occurrences for node $k \in L_2$.
2. *Competition:* Determine the winning neuron *c* and its rival *r* using the rule:

$$d(\bar{x}, \bar{w}_c) = \min_k \left\{d\left(\bar{x}, \bar{w}_k\right)\right\} \qquad d(\bar{x}, \bar{w}_r) = \min_{k \neq c} \left\{d\left(\bar{x}, \bar{w}_k\right)\right\}$$

3. *Adjustment:* Update weight vectors of the winning and the rival neuron according to:

$$\bar{w}_c = \bar{w}_c + \alpha_c \left\|\bar{x} - \bar{w}_c\right\| \qquad \bar{w}_r = \bar{w}_r - \alpha_r \left\|\bar{x} - \bar{w}_r\right\|$$

   where $0 \le \alpha_r \le \alpha_c \le 1$ are the learning rates for the winner and the rival, respectively.
4. If $1/K \sum_{k=1}^{K} \left\|\bar{w}_k(t+1) - \bar{w}_k(t)\right\|^2 < \varepsilon$ then go to step 5. else go to step 1.
5. *Node selection:* Remove all nodes with $n_k = 0$
6. End.

Figure 3. The modified competitive learning algorithm

3.2 Supervised learning phase

After a set of fuzzy rules is extracted from the current training data, i.e. the structure and initial weights of the network are established, the network enters the second learning phase to optimally adjust the parameters based on the same training data (see figure 2). A gradient method performing the steepest descent on a surface in the network weight space is used.

Given the training set, the goal is to adjust weights so as to minimize an overall error function $e = \frac{1}{P} \sum_{p=1}^{P} E_p$ with

$$E_p = \tfrac{1}{2}\sum_{j=1}^{m}\left(y_j^p - f_j^{(3)}\left(\bar{x}^p\right)\right)^2$$

where $f_j^{(3)}\left(\bar{x}^p\right)$ is the jth output of the neuro-fuzzy network for the current sample $\bar{x}^p$, and y_j^p is the corresponding desired output. For the sake of simplicity, the subscript p indicating the current sample will be dropped in the following. The general update formula for a generic weight α is $\Delta\alpha = -\eta\,\partial E/\partial\alpha$ where η is the learning rate. Starting at the first layer, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output values. Then, starting at the output nodes, a backward pass is used to compute $\partial E/\partial\alpha$ for all the nodes. The complete learning algorithm is summarized in Figure 4.

/ Supervised learning */*

1. *Input*: Select the next sample $(\bar{x}.\bar{y})$ from S
2. *Forward step:* propagate $\bar{x}$ through the network and determine the output values $f_j^{(3)},\ j = 1,...,m$
3. *Backward step:* compute error terms for units $j \in L_3$, $k \in L_2$ and $i_k \in L_1$ in the order:

$$\delta_j^{(3)} = -\frac{\partial E}{\partial f_j^{(3)}} = y_j - f_j^{(3)}$$

$$\delta_k^{(2)} = -\frac{\partial E}{\partial f_k^{(2)}} = -\sum_{j=1}^{m}\frac{\partial E}{\partial f_j^{(3)}}\frac{\partial f_j^{(3)}}{\partial f_k^{(2)}} = \sum_{j=1}^{m}\delta_j^{(3)}\left(v_{kj} - f_j^{(3)}\right)$$

$$\delta_{ik}^{(1)} = -\frac{\partial E}{\partial f_{ik}^{(1)}} = -\frac{\partial E}{\partial f_k^{(2)}}\frac{\partial f_k^{(2)}}{\partial f_{ik}^{(1)}} = \delta_k^{(2)}\cdot\frac{\partial f_k^{(2)}}{\partial f_{ik}^{(1)}}$$

4. *Adjustment:* update weights $\{v_{kj}\}, \{w_{ik}\}$ and $\{\sigma_{ik}\}$ respectively according to the update quantities:

$$\Delta v_{kj} = \eta\delta_j^{(3)}\cdot\frac{f_k^{(2)}}{\sum_{t=1}^{K} f_t^{(2)}},\quad \Delta w_{ik} = \eta\delta_{ik}^{(1)}\left[\frac{2(x_i - w_{ik})}{\sigma_{ik}^2}\right]f_{ik}^{(1)},\quad \Delta\sigma_{ik} = \eta\delta_{ik}^{(1)}\left[\frac{2(x_i - w_{ik})^2}{\sigma_{ik}^3}\right]f_{ik}^{(1)}$$

5. If $E < \varepsilon$ then go to step 6. else go to step 1.
6. End.

Figure 4. The supervised learning algorithm

4 Experimental results

The validity of our approach to fuzzy inference and rule extraction has been tested on the well-known benchmark Iris data problem [11]. The classification problem of the Iris data consists of classifying three species of iris flowers (setosa, versicolor and virginica). There are 150 samples for this problem, 50 of each class. A sample is a four-dimensional pattern vector representing four attributes of the iris flower (sepal length, sepal width, petal length, and petal width).

A neuro-fuzzy network architecture with 4 inputs and 3 outputs, corresponding to the 3 classes, was considered.

To estimate the performance of the fuzzy rules extracted by the proposed neuro-fuzzy network, we carried out a 10-fold cross-validation. The whole data set was divided into 10 equally sized parts (15 samples uniformly drawn from the three classes). Each part was used as a test set for the network trained with the remaining 135 data. In each run, an initial number of 10 clusters were considered for the self-organizing phase and the supervised learning procedure was stopped when 100% classification rate was achieved or a maximum number of 1000 epoch.

Table 1 summarizes the classification rate of the 10 neuro-fuzzy networks obtained by the two-stage learning process. As an illustrative example, in Figure 5 the final fuzzy rules extracted by our network are shown for the first trial.

The fuzzy rule bases generated by our approach provide good performance, in terms of classification rate, when compared with other classifiers proposed in literature for the same benchmark problem [12][13][14]. Most of these classifiers were able to predict testing data with the number of misclassified patterns between 2-5. Moreover, it is worth mentioning that most of these results are obtained by computing the apparent misclassification rate, i.e. the error estimated with a one-shot train and test procedure, which is usually an over-optimistic estimate of the actual misclassification rate. In addition, our results outperform those reported in literature in terms of simplicity, providing the smaller number of rules.

		after competitive learning				after supervised learning			
		training set		test set		Training set		Test set	
run	no. rules	classification rate	no. misclass.	classification rate	no. misclass.	Classification rate	no. misclass.	classification rate	no. misclass.
1	5	88,89	15	80,00	3	99,26	1	100,00	0
2	6	91,11	12	86,67	2	99,26	1	93,33	1
3	6	88,89	15	93,33	1	99,26	1	100,00	0
4	9	88,89	15	93,33	1	100,00	0	93,33	1
5	9	88,89	15	86,67	2	100,00	0	86,67	2
6	7	91,11	12	73,33	4	100,00	0	100,00	0
7	7	97,04	4	93,33	1	100,00	0	86,67	2
8	7	88,89	15	93,33	1	100,00	0	100,00	0
9	5	88,15	16	86,67	2	99,26	1	100,00	0
10	8	95,56	6	100,00	0	100,00	0	100,00	0
ave.	6,90	90,74	12,50	88,67	1,70	99,70	0,40	96,00	0,60

Table 1. Results of the 10-fold cross validation.

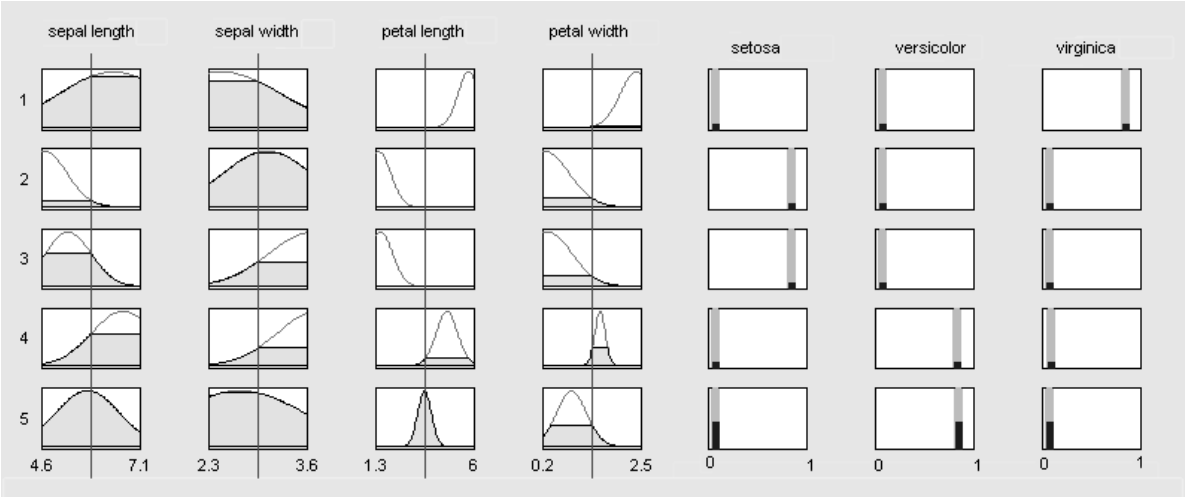

Figure 5. Final fuzzy rules obtained in the first of the 10 trials.

5 Conclusions

In this paper a neural network with a two-phase hybrid learning scheme is proposed to resolve the main problem of fuzzy modeling, i.e. structural and parametric identification of a fuzzy rule base. The main features and advantages of the developed network are:

- it is a general framework that combines two technologies, namely neural networks and fuzzy systems
- the inside of the neural network can be explained in concept of a fuzzy model and hence it can be easily understood
- the network encodes in its structure the essential design parameters to assemble a fuzzy model and processes data according to a fuzzy reasoning mechanism
- the only a-priori information needed from the designer is an initial guessed number of fuzzy rules: the network determines both structure and parameters of the fuzzy rule base via learning from data

Simulations on a well-known classification benchmark verify the effectiveness of the proposed network. We expect that the proposed network, and the underlying approach, should be considered further in respect to a wider range of real-world problems. These applications are the subject of our on-going research.

References

[1] T. Takagi and S.M. Sugeno, Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, and Cybernetics*, vol. 15, 1985, pp. 116-132.

[2] J.S.R. Jang, C.T. Sun, "Neurofuzzy modeling and control," *IEEE Trans. Fuzzy Sys.*, vol. 3, no. 3, March 1995, pp. 378-406.

[3] L.X. Wang and J. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, and Cyb*., vol. 22, 1992, pp. 1414-1427.

[4] D.A. Linkens, M-Y. Chen, "Input selection and partition validation for fuzzy modeling using neural network," *Fuzzy Sets and Systems*, vol. 107, 1999, pp. 299-308.

[5] M. Figueiredo, F. Gomide, "Design of fuzzy systems using neurofuzzy networks," *IEEE Trans. on Neural Networks*, vol. 10, no. 4, July 1999, pp. 815-827.

[6] C. Lin, Y. Lu, "A neural fuzzy system with supervised learning," *IEEE Trans. Syst., Man, and Cyb. - Part B*, vol. 26, 1996, pp. 744-763.

[7] H. Bersini, G. Bontempi, "Now comes the time to defuzzify neuro-fuzzy models," *Fuzzy Sets and Systems*, vol. 90, 1997, pp. 161-169.

[8] C. Lin, C.S.G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, December 1991, pp. 1320-1336.

[9] C-F. Juang, C-T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. on Fuzzy Systems*, vol. 6, no.1, February 1998, pp. 12-32.

[10] L. Xu, A. Krzyzak, E. Oja, "Rival Penalized Competitive Learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, July 1993, pp. 636-649.

[11] R.A. Fisher, "The use of multiple measurements in taxonomic problems, *Ann. Eugen.*, vol. 7, pp. 179-188, 1936.

[12] P.K. Simpson, "Fuzzy min-max neural networks - part I: Classification," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, 1992, pp. 776-786.

[13] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy rule-based classification systems," *IEEE Trans. on Fuzzy Systems*, vol. 4, no. 3, 1996, pp. 238-250.

[14] S. Abe and M. Lan, "A method for fuzzy rules extraction directly from numerical data and its application for function approximation," *IEEE Trans. Syst., Man, and Cyb.* , vol. 25, no. 1, 1996, pp. 744-763.