A Fast Approach for Automatic Generation of Fuzzy Rules by Generalized Dynamic Fuzzy Neural Networks

Shiqian Wu, Meng Joo Er, Member, IEEE, and Yang Gao

Abstract—In this paper, a fast approach for automatically generating fuzzy rules from sample patterns using generalized dynamic fuzzy neural networks (GD-FNNs) is presented. The GD-FNN is built based on ellipsoidal basis function and functionally is equivalent to a Takagi–Sugeno–Kang fuzzy system. The salient characteristics of the GD-FNN are: 1) structure identification and parameters estimation are performed automatically and simultaneously without partitioning input space and selecting initial parameters a priori; 2) fuzzy rules can be recruited or deleted dynamically; 3) fuzzy rules can be generated quickly without resorting to the backpropagation (BP) iteration learning, a common approach adopted by many existing methods. The GD-FNN is employed in a wide range of applications ranging from static function approximation and nonlinear system identification to time-varying drug delivery system and multilink robot control. Simulation results demonstrate that a compact and high-performance fuzzy rule-base can be constructed. Comprehensive comparisons with other latest approaches show that the proposed approach is superior in terms of learning efficiency and performance.

Index Terms—Ellipsoidal basis function (EBF), function approximation, fuzzy rule extraction, on-line self-organizing learning, Takagi–Sugeno–Kang (TSK) fuzzy reasoning.

I. INTRODUCTION

FUZZY logic is a key tool to express knowledge of domain experts so that valuable experience of human beings can be incorporated into controllers design and applied to handle real-life situations that the classical control approach finds difficult or impossible to tackle. Fuzzy systems, as a model-free approach, can approximate any continuous function on a compact set to any accuracy. It has been shown that fuzzy-logic-based modeling and control could serve as a powerful methodology for dealing with imprecision and nonlinearity efficiently [1], [2]. However, the conventional way of designing a fuzzy system has been a subjective approach. It is difficult for a designer, even an expert, to examine all the input–output data from a domain expert, to decide membership functions and the number of fuzzy rules. Recently, more attention has been focused on fuzzy neural networks (FNNs) to acquire fuzzy rules based on the learning ability of neural networks [9], [10]. These FNNs can readily solve two problems of conventional fuzzy reasoning: 1) lack of systematic design for membership functions and 2) lack of adaptability for possible changes in the reasoning environment.

These two problems intrinsically concern parameter estimation. Nevertheless, structure identification, such as partitioning the input and output space and determination of number of fuzzy rules, is still time-consuming. The reason is that, as shown in [10], the problem of determining the number of hidden nodes in neural networks can be viewed as the choice of the number of fuzzy rules. Different from the aforementioned FNNs, several adaptive paradigms have been presented whereby not only can the connection weights be adjusted but also the structure can be self-adaptive during learning [10]. In [11], a hierarchically self-organizing approach, by which the structure was identified by input–output pairs, was developed. An on-line self-constructing paradigm was proposed in [12]. The premise structure in [13] is determined by clustering the input space and generating fuzzy rules. Accordingly, the FNN proposed in [14] is inherently a modified Takagi–Sugeno–Kang (TSK) fuzzy system. As BP learning
will be performed after structure identification, the assignment of initial parameters is not restricted. Based on the ideas of [17] and [18], a hierarchical on-line self-organizing learning algorithm for dynamic fuzzy neural networks (D-FNNs) based on radial basis function (RBF) neural networks, which are functionally equivalent to TSK fuzzy systems, has been developed in [16]. The methodology therein is summarized as follows. The system starts with no rules. Then, rules can be recruited or deleted dynamically according to their significance to system performance so that not only can the parameters be adjusted but also the structure can be self-adaptive. As no iterative learning is employed, the learning speed is very fast. By using the pruning technology, significant nodes are selected so that a parsimonious structure with high performance can be achieved [16]. However, it should be noted that although the D-FNN is a neural-networks-driven fuzzy system, and fuzzy rules can be easily extracted from the D-FNN. When fuzzy rules are acquired from the D-FNN, the following problems exist.

1) All the widths of Gaussian membership functions of the input variables in a rule are the same due to the use of RBF neural networks. This usually does not coincide with the reality, especially when input variables have significantly different operating intervals.

2) The number of membership functions is equal to number of fuzzy rules irrespective of how the membership functions distribute. This results in significant overlapping for some membership functions and are opaque for users to understand.

3) The initial width \( \sigma_0 \) is selected randomly, and the width of a newly generated Gaussian membership function is only determined by the minimal Euclidian distance (E-distance) \( d_{\text{min}} \) and the overlap factor \( k \). If a large value of \( k \) is chosen or if \( d_{\text{min}} \) is large (for instance at the beginning of learning), the widths will be very large.

4) Several prespecified parameters are selected randomly so that it is not easy for users to implement.

Based on the key idea of the D-FNN, a fast approach for automatic generation of fuzzy rules is developed. To be precise, a generalized D-FNN (GD-FNN) based on ellipsoidal basis function (EBF) is presented. This paradigm is based on the fuzzy \( \varepsilon \)-completeness. A novel on-line parameter allocation mechanism is developed to alleviate random choice of initialization. Sensitivities of both input variables and fuzzy rules to the system performance are analyzed so that the width of an input variable in each rule can be adaptive on-line according to its contribution to the system performance. The salient features of the approach can be summarized as follows.

1) Fuzzy rules can be gained quickly without using the BP iteration learning.

2) A hierarchical on-line self-organizing learning is adopted so that structure and parameter identification are done automatically and simultaneously without partitioning the input space and selecting initial parameters a priori.

3) The fuzzy system is constructed rule by rule, i.e., each fuzzy rule is generated according to criteria of rule generation.

4) The number of rules does not increase exponentially with increase in the number of input variables.

5) Human knowledge (in the form of IF-THEN rules) can be directly incorporated into the system.

This paper is organized as follows. Section II presents the architecture of the GD-FNN, which is functionally equivalent to TSK fuzzy systems. Based on the proposed GD-FNN, a learning algorithm is given in details in Section III. Section IV shows the simulation results and some comparative studies with other learning algorithms. Some important issues are discussed in Section V. Lastly, conclusions are drawn in Section VI.

II. ARCHITECTURE OF THE GD-FNN

The structure of the GD-FNN is shown in Fig. 1.\(^1\)

![Fig. 1. Architecture of the GD-FNN.](image)

Let \( r \) be the number of input variables. Each input variable \( x_i \) \( (i = 1, 2, \ldots, r) \) has \( u \) membership function functions \( A_{ij} \) \( (j = 1, 2, \ldots, u) \) shown in level 2, which is in the form of Gaussian functions

\[
\mu_{ij}(x_i) = \exp \left( -\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right) \quad i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, u
\]

(1)

where \( \mu_{ij} \) is the \( j \)th membership function of \( x_i \), \( c_{ij} \) and \( \sigma_{ij} \) are the center and width of the \( j \)th Gaussian membership function of \( x_i \), respectively. If the \( T \)-norm operator used to compute each rule’s firing strength is multiplication, the output of the \( j \)th rule \( R_j \) \( (j = 1, 2, \ldots, u) \) in layer 3 is

\[
\phi_j(x_1, x_2, \ldots, x_r) = \exp \left[ -\sum_{i=1}^{r} \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right] \quad j = 1, 2, \ldots, u
\]

(2)

\(^1\)For notational simplicity, we shall consider multi-input and single-output systems in the following analysis. But the results can be readily applied to multi-input and multi-output systems.
Each node in layer 4 represents an output variable as the weighted summation of incoming signals

$$y(x_1, x_2, \ldots, x_r) = \sum_{j=1}^{u} w_{ij} \cdot \phi_j$$

(3)

where \( y \) is the value of an output variable and \( w_{ij} \) is the THEN-part (consequent parameters) or connection weight of the \( j \)th rule.

For the TSK model, the consequences are the polynomials in the input variables

$$w_j = \alpha_{0j} + \alpha_{1j} x_1 + \cdots + \alpha_{rj} x_r \quad j = 1, 2, \ldots, u$$

(4)

where \( \alpha_{0j}, \alpha_{1j}, \ldots, \alpha_{rj}, j = 1, 2, \ldots, u \) are the weights of input variables in rule \( j \).

Remarks: We have the following observations concerning the structure of the GD-FNN.

1) The firing strength of each rule shown in (2) can be regarded as a function of regularized Mahalanobis distance (M-distance), i.e.,

$$\phi_j = \exp[-\beta M^2(j)]$$

(5)

where

$$M(j) = \sqrt{(X - C_j)^T \Sigma_j^{-1} (X - C_j)}$$

(6)

is the M-distance, where \( X = (x_1 \cdots x_r)^T \in \mathbb{R}^r, C_j = (x_{1j}, x_{2j}, \ldots, x_{rj})^T \in \mathbb{R}^r \) and \( \Sigma_j^{-1} \) is as follows:

$$\Sigma_j^{-1} = \begin{pmatrix} \frac{1}{\sigma_{1j}^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_{2j}^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sigma_{rj}^2} \end{pmatrix} \quad j = 1, 2, \ldots, u$$

(7)

It is obvious that the receptive fields in this model are hyperellipsoidal instead of hypersphere in RBF units.

2) Unlike the models of the D-FNN [16], different input variables \( x_j \) have a different total number of membership functions. In other words, repeated membership function occurs within \( A_{kj} \) (\( j = 1, 2, \ldots, u \)).

3) It has been proven that the model shown in Fig. 1 is a universal approximator [15].

4) The fuzzy rule identification is equivalent to determination of the GD-FNN structure. If we do not have any knowledge about the system, this implies that we cannot choose the GD-FNN a priori. This leads us to develop a learning algorithm that is capable of automatically determining the fuzzy rules.

III. LEARNING ALGORITHM FOR THE GD-FNN

A. Criteria of Rule Generation

1) System Errors: Output error of the GD-FNN with respective to the teaching signals is an important factor in determining whether or not a new rule should be recruited. The error criterion can be described as follows.

For each observation \((X^k, t^k), k = 1, 2, \ldots, n\), where \( n \) is the number of total training data, \( X^k \) is the \( k \)th input vector and \( t^k \) is the \( k \)th desired output, compute the overall GD-FNN output \( \hat{y}^k \) of the existing structure using (2)–(3). Define system error as

$$\|e^k\| = ||t^k - \hat{y}^k||.$$  

(8)

If

$$\|e^k\| > k_e$$  

(9)

a new rule should be considered. Here, \( k_e \) is a predefined threshold that decays during the learning process as follows:

$$k_e = \begin{cases} \epsilon_{\max}, & 1 < k < n/3 \\ \epsilon_{\max} \times \epsilon_{\min}, & n/3 \leq k \leq 2n/3 \\ 2n/3 < k \leq n \end{cases}$$

(10)

where \( \epsilon_{\min} \) is desired accuracy of the output of the GD-FNN; \( \epsilon_{\max} \) is maximum error chosen; \( k \) is learning epoch; \( \beta \in (0, 1) \) is convergence constant, which can be easily deduced as

$$\beta = \left(\frac{\epsilon_{\min}}{\epsilon_{\max}}\right)^{3/n}.$$

(11)

2) \( \varepsilon \)-Completeness of Fuzzy Rules: To facilitate the following development, we formally define the concept of \( \varepsilon \)-completeness of fuzzy rules here.

**Definition 1: \( \varepsilon \)-Completeness of Fuzzy Rules [19]:** For any input in the operating range, there exists at least one fuzzy rule so that the match degree (or firing strength) is no less than \( \varepsilon \).

**Remark:** In fuzzy applications, the minimum value of \( \varepsilon \) is usually selected as \( \varepsilon_{\min} = 0.5 \).

From the viewpoint of fuzzy rules, a fuzzy rule is a local representation over a region defined in the input space. If a new pattern satisfies \( \varepsilon \)-completeness, the GD-FNN will not generate a new rule but accommodate the new sample by updating the parameters of the existing rules.

According to \( \varepsilon \)-completeness, when an observation \((X^k, t^k), k = 1, 2, \ldots, n\), enters the system, we calculate the M-distance \( M^k(j) \) between the observation \( X^k \) and the center \( C_j \) \( (j = 1, 2, \ldots, u) \) of the existing EBF units according to (6)–(7).

Find

$$J = \arg \min_{1 \leq j \leq u} (M^k(j))$$

(12)

If

$$\min_{k} M^k(J) > k_d$$

(13)

where \( k_d \) is a prespecified threshold that decays during the learning process, this implies that the existing system is
not satisfied with $\varepsilon$-completeness and a new rule should be considered. Here, the $k_d$ is chosen as follows:

$$
k_d = \begin{cases} 
\frac{d_{\text{max}}}{\ln(1/\varepsilon)}, & 1 < k < n/3 \\
\max\{d_{\text{max}} \times \gamma^k, d_{\text{min}}\}, & n/3 \leq k \leq 2n/3 \\
\frac{1}{\ln(1/\varepsilon)}, & 2n/3 < k \leq n 
\end{cases} \tag{14}
$$

where $n$ is the learning epoch and $\gamma \in (0,1)$ is called decay constant, which can be calculated by (15)

$$
\gamma = \left(\frac{d_{\text{min}}}{d_{\text{max}}}\right)^{\frac{1}{2}} = \sqrt{\frac{\ln(1/\varepsilon_{\text{max}})}{\ln(1/\varepsilon_{\text{min}})}}, \tag{15}
$$

The key idea for the choice of $k_\varepsilon, k_d$ in the form of (10) and (14) is to first find and cover the most troublesome positions, which have large errors between the desired and actual outputs but are not properly covered by existing rules. This is called coarse learning. When $k_\varepsilon, k_d$ reach $c_{\text{min}}$ and $d_{\text{min}}$, respectively, fine learning begins. This is the idea called hierarchical learning [16], [17].

### B. Estimation of Premise Parameters

After a rule has grown, the problem is how to allocate its parameters. Before elaborating on the details of how to allocate premise parameters, we first introduce the following definition.

**Definition 2: Semiclosed Fuzzy Sets:** Let $U = [a, b]$ be the universe of discourse of input $x$. If each fuzzy set $A_i = \{\mu_i(x)| x \in U\}$ ($i = 1, 2, \ldots, m$) is represented as Gaussian function shown in (1) for all $x \in U$ and satisfies the following boundary conditions simultaneously:

$$
\mu_i(x) = e^{-\frac{(x-c_i)^2}{\sigma_i^2}} \quad \text{if} \quad |c_i - a| \leq \delta \tag{16}
$$

$$
\mu_i(x) = e^{-\frac{(x-b)^2}{\sigma_i^2}} \quad \text{if} \quad |c_i - b| \leq \delta \tag{17}
$$

$$
\mu_i(x) = e^{-\frac{(c_i-a)^2}{\sigma_i^2}} \quad \text{if} \quad |c_i - a| > \delta \quad \text{and} \quad |b-c_i| > \delta \tag{18}
$$

where $\delta$ is a tolerable small value and the widths of Gaussian functions are selected as

$$
\sigma_i = \frac{\max\{|c_i - c_i-1|, |c_i - c_{i+1}|\}}{\ln(1/\varepsilon)} \quad i = 1, 2, \ldots, m \tag{19}
$$

where $c_i-1$ and $c_{i+1}$ are the two centers of neighboring membership functions of $i$th membership function. Then, we call $A_i$ ($i = 1, 2, \ldots, m$) semiclosed fuzzy sets [12].

**Theorem 1:** The semiclosed fuzzy sets $A_i$ ($i = 1, 2, \ldots, m$) in Definition 2 satisfy $\varepsilon$-completeness of fuzzy rules, i.e., for all $x \in U$, there exists $i \in \{1, 2, \ldots, m\}$, such that $\mu_i(x) \geq \varepsilon$.

**Proof:** We can prove this theorem under several different cases.

1) If there exists only one fuzzy set, i.e., $m = 1$, the membership function can be generated according to Definition 2 as follows.

If $|c_1 - a| \geq \delta$ and $|b - c_1| \geq \delta$, and assuming that $|c_1 - a| \geq |c_1 - b|$, then we have

$$
\mu_1(x) = e^{-\frac{(c_1-a)^2}{\sigma_1^2}}
$$

where

$$
\sigma_1 = \frac{|c_1 - a|}{\sqrt{\ln(1/\varepsilon)}}.
$$

For any $x \in [a, b]$, we have

$$
\mu_1(x) \geq \mu_1(a) = \exp\left[-\left(\frac{1}{\varepsilon}\right)\left(\frac{a-c_1}{c_1-a}\right)^2\right] = \varepsilon.
$$

If $|c_1 - a| \leq \delta$, we have

$$
\mu_1(x) = e^{-\frac{(c_1-a)^2}{\sigma_1^2}}
$$

where

$$
\sigma_1 = \frac{|a - b|}{\sqrt{\ln(1/\varepsilon)}}.
$$

For any $x \in [a, b]$, we have

$$
\mu_1(x) \geq \mu_1(b) = \exp\left[-\left(\frac{1}{\varepsilon}\right)\left(\frac{b-a}{a-b}\right)^2\right] = \varepsilon.
$$

2) If there exists $i \in \{1, 2, \ldots, m\}$, such that $|c_i - c_{i-1}| \geq |c_{i+1} - c_{i-1}|$, then the $\sigma_i$ is chosen as

$$
\sigma_i = \frac{|c_i - c_{i-1}|}{\sqrt{\ln(1/\varepsilon)}}.
$$

For any $x \in (c_{i-1}, c_{i+1})$, we have

$$
\mu_i(x) \geq \mu_i(c_{i-1}) = \exp\left[-\left(\frac{1}{\varepsilon}\right)\left(\frac{c_{i-1} - c_i}{c_i - c_{i-1}}\right)^2\right] = \varepsilon.
$$

We can obtain the same result for other cases. This completes the proof.

**Remark:** Definition 2 does not indicate that two membership functions are necessary at the starting point and the ending point. They are only treated as two constraints.

Assume that $u$ fuzzy rules have been generated. A new rule will be formed when the input pattern $x^k$ ($k = 1, 2, \ldots, n$) enters the system according to criteria of rules generation. Next, the incoming multidimensional input vector $X^k$ is projected to the corresponding one-dimensional membership function for each input variable $i$ ($i = 1, 2, \ldots, r$) and compute the E-distance $e_{d_i}$ between the data $x_i^k$ and boundary set $\Phi_i$

$$
e_{d_i}(j) = |x_i^k - \Phi_i(j)| \quad j = 1, 2, \ldots, u + 2 \tag{20}
$$

where $\Phi_i \in \{x_{i_{\text{min}}}, c_1, c_2, \ldots, c_i, x_{i_{\text{max}}}\}$ and find

$$
j_n = \arg\min_{j=1,2,\ldots,u+2}(e_{d_i}(j)). \tag{21}
$$

If

$$
e_{d_i}(j_n) \leq \kappa_{\text{inf}} \tag{22}
$$
where $k_{\text{inf}}$ is a predefined constant that controls the similarity of neighboring membership function, we assume that $x_k^j$ can be completely represented by the existing fuzzy set $A_{j_0}$ ($c_{j_0}, \alpha_{j_0}$) without generating a new membership function. Otherwise, a new Gaussian membership function is allocated whose width is determined by (19) and the center is set as follows:

$$c_{i(u+1)} = x_i^j.$$  \hspace{1cm} (23)

### C. Sensitivity of Input Variables and Fuzzy Rules

1) Error Reduction Ratio: Given $n$ input–output pairs $(X(k), t(k), k = 1, 2, \ldots, n)$, consider (3) as a special case of the linear regression model $t(k) = \sum_{j=1}^{u} w_j(k) \cdot \phi_j(k) + e(k)$, or in the following compact form:

$$D = H\theta + E$$  \hspace{1cm} (24)

where $D \in \mathbb{R}^n$ is the desired output; $H = \phi^T = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^{n \times v}$ are the regressors, with $v = u \times (r+1)$; $\theta = W^T \in \mathbb{R}^u$ contains real parameters; and $E \in \mathbb{R}^n$ is the error vector that is assumed to be uncorrelated with the regressors $h_i (i = 1, 2, \ldots, v)$.

For matrix $H$, if its row number is larger than the column number, we can transform $H$ into a set of orthogonal basis vectors by QR decomposition

$$H = PN$$  \hspace{1cm} (25)

where $P = (p_1, p_2, \ldots, p_v) \in \mathbb{R}^{n \times v}$ has the same dimension as $H$ with orthogonal columns and $N \in \mathbb{R}^{v \times v}$ is an upper triangular matrix. This transformation makes it possible to calculate individual contributions to the desired output energy from each basis vector.

Substituting (25) into (24) yields

$$D = P N \theta + E = P G + E.$$  \hspace{1cm} (26)

The linear least square (LLS) solution of $G$ is given by $G = (P^T P)^{-1} P^T D$, or

$$g_i = \frac{P_i^T D}{P_i^T P_i} \quad i = 1, 2, \ldots, v.$$  \hspace{1cm} (27)

The quantities $G$ and $\theta$ satisfy the following equation:

$$N \theta = G.$$  \hspace{1cm} (28)

As $p_i$ and $p_j$ are orthogonal for $i \neq j$, the sum of squares or energy of $D$ is given as follows [20]:

$$D^T D = \sum_{i=1}^{v} g_i^2 P_i^T P_i + E^T E.$$  \hspace{1cm} (29)

If $D$ is the desired output vector after its mean has been removed, the variance of $D$ is given by

$$n^{-1} D^T D = \sum_{i=1}^{v} g_i^2 P_i^T P_i + n^{-1} E^T E.$$  \hspace{1cm} (30)

It is seen that $\sum_{i=1}^{v} g_i^2 P_i^T P_i / n$ is the part of the desired output variance that can be explained by the regressor $P_i$ and $E^T E / n$ is the unexplained variance of $D$. Thus, $\sum_{i=1}^{v} g_i^2 P_i^T P_i / n$ is the increment to the explained desired output variance introduced by $P_i$, and an error reduction ratio (ERR) due to $P_i$ can be defined as

$$\text{err}_{i} = \frac{g_i^2 P_i^T P_i}{D^T D} \quad i = 1, 2, \ldots, v.$$  \hspace{1cm} (31)

Substituting $g_i$ by (27), we have

$$\text{err}_{i} = \frac{(P_i^T D)^2}{P_i^T P_i D^T D} \quad i = 1, 2, \ldots, v.$$  \hspace{1cm} (32)

The above equation offers a simple and effective means of seeking a subset of significant regressors.

2) Sensitivity of Fuzzy Rules: Define the ERR matrix $\Delta = (\rho_1, \rho_2, \ldots, \rho_u) \in \mathbb{R}^{(r+1) \times u}$ whose elements are obtained from (32) and the $j$th column of $\Delta$ as the total ERR corresponding to the $j$th rule. Furthermore, define

$$\eta_j = \sqrt{\rho_j / r + 1} \quad j = 1, 2, \ldots, u$$  \hspace{1cm} (33)

then $\eta_j$ represents the significance of the $j$th rule. If

$$\eta_j < k_{\text{err}} \quad j = 1, 2, \ldots, u$$  \hspace{1cm} (34)

where $k_{\text{err}}$ is a prespecified threshold, then the $j$th rule is deleted.

3) Sensitivity of Input Variables: Define

$$B_j = \sum_{k=2}^{r+1} \rho_j(k) \quad j = 1, 2, \ldots, u$$  \hspace{1cm} (35)

$$B_{ij} = \frac{\text{err}_{ij}}{B_j} \quad i = 1, 2, \ldots, r$$  \hspace{1cm} (36)

where $B_j$ represents the total ERR related to input variables in the $j$th rule and $\text{err}_{ij}$ is the ERR corresponding to the $i$th input variable in the $j$th rule. Accordingly, $B_{ij}$ indicates the significance of the $i$th input variable in the $j$th rule. If $B_{ij}$ is small, that implies that the $i$th input variable is not sensitive to the system output. Consequently, this idea reveals that the width of the hyperellipsoidal region in $i$th input variable could be reduced without significant effect of system performance.

### D. Width Modification

In Section III-A, we propose that a new rule should be generated in the case where $||e^k|| > k_e$ and $\kappa^{k}_{\text{inf}} > k_d$. Yet another important case where $||e^k|| > k_e$ and $\kappa^{k}_{\text{inf}} \leq k_d$ needs to be considered because the performance is not good. This case indicates that although $X^k$ can be accommodated by the adjacent fuzzy rule, the significance of the rule is not so great as to accommodate all the patterns covered by the EBF unit, or the system outputs vary sharply in this ellipsoidal region. Accordingly, the ellipsoidal region should be decreased to obtain a better local approximation.

For an incoming sample $X^k$, we can find the $j$th rule that is “nearest” to it by M-distance. If the conditions $||e^k|| > k_e$ and $\kappa^{k}_{\text{inf}} \leq k_d$ are satisfied, we decompose the vector $X^k$ to
Fig. 2. Flowchart of the learning algorithm for the GD-FNN.

the corresponding axis of input variables. Then the width of the nearest membership function \(j\) \((=1,2,\ldots,u)\) of input variable \(i\) \((=1,2,\ldots,r)\) is modified as

\[
\sigma_{ij}^{\text{new}} = \zeta \times \sigma_{ij}^{\text{old}}
\]  \hspace{1cm} (37)

where \(\zeta \in (0,1)\) is a reduced factor that is determined based on sensitivity of input variables as follows:

\[
\zeta = \begin{cases} 
1 & B_{ij} < 1/r \\
\frac{1}{1+k_w(B_{ij} - 1/r)^2} & B_{ij} \geq 1/r 
\end{cases}
\]  \hspace{1cm} (38)
where \( r \) is total number of input variables. Equation (38) implies that if the ERR of the \( i \)th input variable is less than the average level in all terms related to input variables in the \( j \)th rule, the width of the \( i \)th input variable will be decreased. If we set \( \zeta_{\text{min}} = k_s \) when \( B_{ij} = 0 \), then the value of parameter \( k_w \) can be obtained from (38) and the final version for \( \zeta \) is as follows:

\[
\zeta = \begin{cases} 
  k_s \\
  k_s + r^2(1 - k_s)(B_{ij} - 1/r)^2 \\
  1 
\end{cases} \quad \text{for } \begin{cases} 
  B_{ij} < 1/r \\
  B_{ij} \geq 1/r 
\end{cases} 
\]  

(39)

E. Determination of Consequent Parameters

Suppose that \( n \) fuzzy rules are generated for \( n \) observations with \( r \) number of input variables. Rewriting (3) in matrix form yields

\[
W \phi = Y 
\]  

(40)

where

\[
W \in \mathbb{R}^{n \times (r+1)}; \\
\phi \in \mathbb{R}^{(r+1) \times n}; \\
Y \in \mathbb{R}^n.
\]

Assume that the desired output is \( T = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n \). The problem of determining the optimal parameters \( W^* \) can be formulated as a linear problem of minimizing \( \|W \phi - T\|_2 \) and \( W^* \) is determined by the pseudoinverse technique [16]

\[
W^* = T(\phi^T \phi)^{-1} \phi^T 
\]  

(41)

where \( \phi^T \) is the transpose of \( \phi \) and \( \phi^+ = (\phi^T \phi)^{-1} \phi^T \) is the pseudoinverse of \( \phi \).

The flowchart of the algorithm is illustrated in Fig. 2.
In this section, the effectiveness of the proposed algorithm is demonstrated in function approximation, nonlinear dynamic system identification, time-varying drug delivery system, and multilink robot control. Some comparisons are made with other earlier work such as fuzzy model (FM) [3], NN-driven fuzzy reasoning (NNDFR) [10], fuzzy-neural systems using similarity analysis (FNS) [12], adaptive-network-based fuzzy inference systems (ANFISs) [13], D-FNN [16], RBF-based adaptive fuzzy systems (RBF-AFSs) [17], orthogonal least squares (OLS) [20], and so on.

**Example 1—Modeling a Three-Input Nonlinear Function:** In this example, the underlying function to be approximated is a three-input nonlinear function given by

\[(42)\]

which is widely used to verify the approaches adopted in [3], [10], and [13].

A total of 216 training data are randomly sampled from the input ranges \([1,6] \times [1,6] \times [1,6]\). The parameters are selected as follows:

- \(\varepsilon_{\text{min}} = 0.5\)
- \(\varepsilon_{\text{max}} = 0.8\)
- \(k_{\text{inf}} = 0.65\)
- \(k_{\text{uf}} = 0.9\)
- \(k_{\text{err}} = 0.002\)

The results are shown in Fig. 3, in which we see that there are ten fuzzy rules. The input variables \(x, y,\) and \(z\) define three, four, and five membership functions, respectively, which are shown in Fig. 4. The ten fuzzy rules are shown in Table I.

To compare the performance, the same performance index adopted in [3], [10], and [13] is used. For the convenience of the readers, the performance index is reproduced here

\[(43)\]

where \(n\) is the number of data pairs and \(t(i)\) and \(y(i)\) are the \(i\)th desired output and calculated output, respectively. Another 125 data are randomly selected from the same operating range to check the generalization of the learned GD-FNN. Comparisons of the GD-FNN with FM, NNDFR, ANFIS, and OLS are shown in Table II.

We see from Table II that the GD-FNN has great larger training error than the ANFIS, although it has more parameters. The reason is that the ANFIS is implemented by the BP iterative learning so that the optimal parameters could be achieved. For the GD-FNN, the fast learning speed is achieved at the expense that the overall GD-FNN does not offer optimal solutions for training patterns because no learning is performed on the overall system. However, it is worth mentioning that the GD-FNN possesses good generation because it provides comparable testing error to the ANFIS.

**Example 2—Nonlinear Dynamic System Identification:** The plant to be identified is described as

\[(44)\]

which is also used in [1], [12], [16], and [17]. To identify the plant, a series-parallel identification model governed by the fol-

<table>
<thead>
<tr>
<th>Number of fuzzy rules</th>
<th>Premise Parameters</th>
<th>Consequent parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(X)</td>
<td>(Y)</td>
</tr>
<tr>
<td>1</td>
<td>((4.2, 3.5))</td>
<td>((1.0, 4.2))</td>
</tr>
<tr>
<td>2</td>
<td>((2.2, 2.2))</td>
<td>((6.0, 4.8))</td>
</tr>
<tr>
<td>3</td>
<td>((2.2, 2.2))</td>
<td>((3.7, 3.2))</td>
</tr>
<tr>
<td>4</td>
<td>((6.0, 1.9))</td>
<td>((1.0, 4.2))</td>
</tr>
<tr>
<td>5</td>
<td>((4.2, 3.5))</td>
<td>((1.0, 4.2))</td>
</tr>
<tr>
<td>6</td>
<td>((6.0, 1.9))</td>
<td>((6.0, 4.8))</td>
</tr>
<tr>
<td>7</td>
<td>((6.0, 1.9))</td>
<td>((2.5, 1.8))</td>
</tr>
<tr>
<td>8</td>
<td>((4.2, 3.5))</td>
<td>((6.0, 4.8))</td>
</tr>
<tr>
<td>9</td>
<td>((4.2, 3.5))</td>
<td>((1.0, 4.2))</td>
</tr>
<tr>
<td>10</td>
<td>((4.2, 3.5))</td>
<td>((6.0, 4.8))</td>
</tr>
</tbody>
</table>

* \(A\) indicates a Gaussian membership function. The first value and the second value in parentheses represent the center and the width of the corresponding Gaussian function respectively.
The following equation is used:

$$
\hat{y}(t + 1) = f(y(t), y(t - 1), u(t))
$$

(45)

where $f$ is the function implemented by the GD-FNN with three inputs and one output. The input has the form

$$
u(t) = \sin(2\pi t/25).
$$

(46)

Selecting the parameters as follows:

$$
\varepsilon_{\text{min}} = 0.5, \quad \varepsilon_{\text{max}} = 0.8, \quad \varepsilon_{\text{min}} = 0.03, \quad \varepsilon_{\text{max}} = 0.5, \\
k_{\text{init}} = 0.5, \quad k_s = 0.9 \quad \text{and} \quad k_{\text{err}} = 0.003.
$$

The simulation results are shown in Fig. 5, and membership functions of the input variable $y(t)$, $y(t-1)$ and $u(t)$ are shown in Fig. 6.

The fuzzy rules are listed as follows:

**Rule 1** If $y(t)$ is $A(0.4,1)$ and $y(t-1)$ is $A(0,4.1)$ and $u(t)$ is $A(0,1.1)$, then $y(t+1) = -0.77 + 3.06y(t) - 0.80y(t-1) + 0.91u(t)$.

**Rule 2** If $y(t)$ is $A(0.8,2.1)$ and $y(t-1)$ is $A(0,6.9)$ and $u(t)$ is $A(1,1)$, then $y(t+1) = 4.38 + 4.12y(t) + 4y(t-1) - 2.25u(t)$.

**Rule 3** If $y(t)$ is $A(2.3,2.3)$ and $y(t-1)$ is $A(1.8,3.1)$ and $u(t)$ is $A(1,1)$, then $y(t+1) = -27.8 + 3.08y(t) + 3.75y(t-1) + 11.44u(t)$.

**Rule 4** If $y(t)$ is $A(3.7,1.4)$ and $y(t-1)$ is $A(3.7,2.1)$ and $u(t)$ is $A(0,1.1)$, then $y(t+1) = 37.86 - 22.50y(t) + 11.75y(t-1) + 29.34u(t)$.

**Rule 5** If $y(t)$ is $A(-1,6,0.97)$ and $y(t-1)$ is $A(0,3.7)$ and $u(t)$ is $A(-1,0.97)$, then $y(t+1) = 4.85 - 1.51y(t) - 0.05y(t-1) + 5.48u(t)$.

**Rule 6** If $y(t)$ is $A(3.7,1.4)$ and $y(t-1)$ is $A(3.7,2.1)$ and $u(t)$ is $A(1,1)$, then $y(t+1) = -43.72 + 15.58y(t) - 6.35y(t-1) - 2.86u(t)$.

Performance comparisons of different approaches are shown in Table III. We see that the GD-FNN provides high performance on a compact rule base.

<table>
<thead>
<tr>
<th>Model</th>
<th>$APE_{\text{in}}$ (%)</th>
<th>$APE_{\text{cha}}$ (%)</th>
<th>Parameter number</th>
<th>Training set size</th>
<th>Checking set size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM1</td>
<td>1.5</td>
<td>2.1</td>
<td>22</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>FM2</td>
<td>0.59</td>
<td>3.4</td>
<td>32</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>NNDFR</td>
<td>3.51</td>
<td>54.9 ($\varepsilon_m = 4.76$)</td>
<td>–</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>ANFIS</td>
<td>0.043</td>
<td>1.066</td>
<td>50</td>
<td>216</td>
<td>125</td>
</tr>
<tr>
<td>OLS</td>
<td>2.43</td>
<td>2.56 ($\varepsilon_m = 1.3949$)</td>
<td>66</td>
<td>216</td>
<td>125</td>
</tr>
<tr>
<td>GD-FNN</td>
<td>2.11</td>
<td>1.54 ($\varepsilon_m = 0.8781$)</td>
<td>64</td>
<td>216</td>
<td>125</td>
</tr>
</tbody>
</table>

* The sample patterns are mixed with noise in this case.

* $\varepsilon_m$ indicates the maximum difference between the desired output and the calculated output.

**Example 3—Time-Varying Drug Delivery Control System:** Hypertension after a cardiac operation is a well-known phenomenon. Continuous infusion of sodium nitroprusside (SNP) would quickly lower the blood pressure in most patients. An overdose of SNP could cause toxic side effects. It is therefore necessary to control the infusion rate of SNP carefully to achieve the desired blood pressure. However, physiological systems are highly complex in structure. An empirical dynamic model of a patient’s mean arterial pressure (MAP) response to SNP infusion was proposed in [21] as follows:

$$
y(k) = a_0y(k-1) + b_0u(k-d) + b_1u(k-m) \quad (47)
$$

where $y(k)$ and $u(k)$ are the change in MAP from the initial blood pressure and infusion rate of SNP at discrete time $k$, respectively. The parameters $a_0, b_0, b_1, d,$ and $m$ vary from a patient to another. A range of typical values for these parameters is listed in Table IV.

In this example, the control scheme for the infusion rate of SNP by utilizing the GD-FNN is shown in Fig. 7.

Assume that the parameters and the structure are changed as follows (the parameters vary in the same way as [22], but the structure in [22] is unchanged)

$$
a_0 = \begin{cases} 
0.4 & 0 \leq k < 30 \\
0.6 + 0.005(k - 150)/3 & 30 \leq k < 90 \\
0.6 + 0.01(k - 120)/3 & 90 \leq k < 120 \\
0.6 & k \geq 200
\end{cases}
$$

$$
b_0 = \begin{cases} 
2.4 + 0.008(k - 200)/3 & k \leq 200 \\
2.4 & k > 200
\end{cases}
$$

$$
b_1 = \begin{cases} 
0.96 + 0.0032(k - 200)/3 & k \leq 200 \\
0.96 & k > 200
\end{cases}
$$

$$
d = \begin{cases} 
3 & 0 \leq k \leq 100 \\
4 & 100 \leq k \leq 200
\end{cases}
$$

$$
m = \begin{cases} 
6 & 0 \leq k \leq 100 \\
8 & 100 \leq k \leq 200.
\end{cases}
$$
Then the inverse model of (47) is represented by the GD-FNN as follows:

\[ \hat{u}(k) = f(y(k), y(k-3)). \]  

(48)

If the output \( y(t) \) is corrupted by white noise with variance of 1 mm Hg, then the desired and actual infusion rate is shown in Fig. 8.

The performance of the GD-FNN controller is evaluated in terms of the maximum error \( \delta_{\text{max}} \) between the desired and actual change of MAP. The comparison with the result of [22] (under the same noisy environment) is listed in Table V.

Remarks: We choose inverse control scheme because the motivation of this example is to verify whether the GD-FNN could approximate a time-varying system. In general, the absence of feedback in inverse control always results in lack of robustness and, accordingly, other control strategies need to be considered if we wish to achieve better performance. The results shown in Fig. 8 and Table V demonstrate that the GD-FNN can exactly approximate the time-varying system.

Example 4—On-Line Manipulator Control: In this example, the GD-FNN is applied to a multilink robot control problem. The control structure is illustrated in Fig. 9. The basic idea of this approach is to learn the manipulator’s inverse characteristics and use the inverse dynamic model to generate the compensated control signal. The inverse robot model is obtained by FNN A employing the GD-FNN learning algorithm. FNN A is trained during real-time control of the manipulator and is expected to have high adaptability and flexibility. FNN B is simply a duplicate copy of FNN A, but its weights will be further adjusted by the error torque signal to produce the desired torque to drive the manipulator. This is to compensate for modeling errors. FNN B is connected in parallel with a conventional PD controller, and very fast and accurate tracking performance can be achieved by virtue of this compensation.

Simulation studies were carried out to verify that the proposed fuzzy neural controller could compensate for unmodeled disturbances. The manipulator used for the simulation study is a typical two-degrees-of-freedom robot depicted in Fig. 10.

The dynamic equation of the manipulator is generally given by

\[ M(\dot{q})\ddot{q} + Q(q, \dot{q}) + \tau_d = \tau = [\tau_1 \ \tau_2]^T \]  

(49)

where

- \( M \) 2 × 2 inertia matrix of the manipulator;
- \( Q \) 2 × 1 vector of centrifugal, Coriolis, friction forces, and gravity;
- \( \tau_d \) 2 × 1 vector of unknown terms arising from unmodeled dynamics and external disturbances;
- \( \tau \) 2 × 1 vector of the input torque generated by the joint motor.

The dynamic equation of the proposed manipulator is further derived from [23] as shown in (50) at the bottom of the next
Fig. 6. Membership functions of the input variables (a) $y(t)$, (b) $y(t-1)$, and (c) $u(t)$.

...where the parameters of the two-link planar manipulator used for the simulation are as follows:

- $m_1$: mass of link 1 = 1 kg;
- $L_1$: length of link 1 = 1 m;
- $m_e$: mass of link 2 and payload = 2 kg;
- $\delta_e$: angle of payload with respect to link 2 = 30°;
- $I_1$: centroidal moment of inertia of link 1 = 0.12 kg.m$^2$;
- $L_{c1}$: length of center of gravity (CG) of link 1 from the axis of rotation = 0.5 m;
- $I_e$: centroidal moment of inertia of link 2 and payload = 0.25 kg.m$^2$;
- $L_{ce}$: length of CG of link 2 and payload from the axis of rotation = 0.6 m

In the simulated trajectory control of the manipulator by the fuzzy neural controller, the desired trajectories in terms of link angular positions were chosen to be $1.5\sin(2\pi t)$ rad and $0.7\cos(2\pi t)$ rad for $q_1$ and $q_2$, respectively. The initial angular position and velocity of both links were all set to zero. The external disturbance $\tau_{d1}$ and $\tau_{d2}$ were assumed to be $100\sin(2\pi t)$ Nm and $50\sin(2\pi t)$ Nm, which are comparable to the magnitude of control torque for the manipulator. The proportional gain $K_p$ and the differential gain $K_v$ of the conventional PD controller were set at 25 and 7, respectively.

\[\begin{align*}
\tau_1 &= [(I_1 + m_1L_{c1}^2 + I_e + m_eL_{ce}^2 + m_eL_1^2) \\
&\quad + 2(m_eL_{c1} \cos \delta_e) \cos q_2 + 2(m_eL_{ce} \sin \delta_e) \sin q_2] \dot{q}_1 \\
&\quad + [(I_1 + m_eL_{c1}^2) + (m_eL_{ce} \cos \delta_e) \cos q_2 + (m_eL_{ce} \sin \delta_e) \sin q_2] \dot{q}_2 \\
&\quad - [(m_eL_{ce} \cos \delta_e) \sin q_2 - (m_eL_{ce} \sin \delta_e) \cos q_2] \dot{q}_1 \dot{q}_2 \\
&\quad - [(m_eL_{ce} \cos \delta_e) \sin q_2 - (m_eL_{ce} \sin \delta_e) \cos q_2] (\dot{q}_1 + \dot{q}_2)^2 + \tau_{d1}
\end{align*}\]

\[\begin{align*}
\tau_2 &= [(I_1 + m_eL_{ce}^2) + (m_eL_{ce} \cos \delta_e) \cos q_2 \\
&\quad + (m_eL_{ce} \sin \delta_e) \sin q_2] \dot{q}_1 + (I_e + m_eL_{ce}^2) \dot{q}_2 \\
&\quad + [(m_eL_{ce} \cos \delta_e) \sin q_2 - (m_eL_{ce} \sin \delta_e) \cos q_2] \dot{q}_1^2 + \tau_{d2}
\end{align*}\]
TABLE III
PERFORMANCE COMPARISONS OF DIFFERENT APPROACHES

<table>
<thead>
<tr>
<th></th>
<th>Fuzzy model [1]</th>
<th>FNS</th>
<th>RBF-AFS</th>
<th>OLS</th>
<th>D-FNN</th>
<th>GD-FNN (case 1)</th>
<th>GD-FNN (case 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rules</td>
<td>40</td>
<td>22</td>
<td>35</td>
<td>65</td>
<td>6</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>200</td>
<td>84</td>
<td>280</td>
<td>326</td>
<td>48</td>
<td>48</td>
<td>56</td>
</tr>
<tr>
<td>RMSE *</td>
<td>--</td>
<td>**</td>
<td>0.1384</td>
<td>0.0288</td>
<td>0.0283</td>
<td>0.0241</td>
<td>0.0108</td>
</tr>
</tbody>
</table>

*RMSE—Root Mean Squared Error. ** The results are not listed in the original papers.

TABLE IV
RANGE OF VALUES FOR THE PARAMETERS $a_0$, $b_0$, $b_1$, $d$, and $m$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Nominal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>0.606</td>
<td>0.779</td>
<td>0.741</td>
</tr>
<tr>
<td>$b_0$</td>
<td>0.053</td>
<td>3.547</td>
<td>0.187</td>
</tr>
<tr>
<td>$b_1$</td>
<td>0</td>
<td>1.418</td>
<td>0.075</td>
</tr>
<tr>
<td>$d$</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$m$</td>
<td>4</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

Fig. 7. Inverse control scheme of the drug delivery control system.

Fig. 8. Time-varying drug delivery control system. (a) Actual infusion rate of SNP under noisy environment. (b) Actual change of MAP under noisy environment [actual (—) and desired (—) change].
The tracking error $E$ is defined as follows:

$$E = (q_{d1} - q_1)^2 + (q_{d2} - q_2)^2$$  \hspace{1cm} (51)$$

where $q_{d1}$ and $q_{d2}$ are the desired trajectories on the $q_1$-$q_2$ plane at the $i$th sampling period and $q_1$ and $q_2$ are the actual trajectories, respectively.

The FNN controller learns the manipulator dynamics with external disturbances on-line. Fig. 11 shows the simulation results of trajectory control by the proposed controller. The figure shows the first-trial, second-trial results, and results from the third trial onwards of drawing an ellipse on the $q_1$-$q_2$ plane. It took about 1 s for one trial, and the sampling period used was 1 ms. The results clearly demonstrate that the two links, whose initial positions were at the origin, were able to track the desired trajectories from the third trial onwards. The performance of the control system is greatly improved after adding the GD-FNN to

<table>
<thead>
<tr>
<th>Technique</th>
<th>$dp_{\text{max}}$ (mm Hg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IANC [22]</td>
<td>10</td>
</tr>
<tr>
<td>GD-FNN</td>
<td>9.08</td>
</tr>
</tbody>
</table>

TABLE V
THE GD-FNN PERFORMANCE UNDER NOISEY ENVIRONMENT
TABLE VI

<table>
<thead>
<tr>
<th>Controller</th>
<th>Steady State Error (rad)</th>
<th>Overshoot (%)</th>
<th>Rise Time (sec)</th>
<th>Maximum Control Torque (Nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD</td>
<td>$q_1$</td>
<td>0.050</td>
<td>9.3</td>
<td>0.287</td>
</tr>
<tr>
<td>$K_p=6000; K_v=100$</td>
<td>$q_2$</td>
<td>0.018</td>
<td>7.1</td>
<td>0.254</td>
</tr>
<tr>
<td>AFC</td>
<td>$q_1$</td>
<td>0.022</td>
<td>0</td>
<td>0.237</td>
</tr>
<tr>
<td>$K_p=25; K_v=35$</td>
<td>$q_2$</td>
<td>0.025</td>
<td>0</td>
<td>0.218</td>
</tr>
<tr>
<td>FNC</td>
<td>$q_1$</td>
<td>0.001</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>$K_p=25; K_v=7$</td>
<td>$q_2$</td>
<td>0.001</td>
<td>0</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Fig. 12. Convergence of tracking error.

the conventional PD controller. It can also be seen from Fig. 12 that the tracking error converges to zero very quickly.

A comparison of the transient and steady-state performance of the proposed fuzzy neural controller (FNC) compared with a PD controller and a newly developed adaptive fuzzy controller (AFC) of [24] is tabulated in Table VI. This time, the control objective is to draw a straight line passing the origin on the $q_1$-$q_2$ plane. From Table VI, we can see that the FNC has the best performance among the three control systems in terms of transient and steady-state results.

V. DISCUSSIONS

The motivation of this paper is to provide a simple and fast approach to configure a fuzzy system so that 1) some meaningful fuzzy rules could be acquired for knowledge engineering and 2) the system could be used as a system modeling tool for control engineering, pattern recognition, and so on. Many researchers have addressed these issues and developed different paradigms, as shown in Section I. Here, we would like to provide more insights into these algorithms and compare with the proposed GD-FNN.

A. Input–Output Space Partition

This is one of the key issues in fuzzy systems because it determines the structure of a fuzzy system. The common approach of the conventional input space partitioning is the so-called grid-type partitioning. The limitation of this partition is the “curse of dimensionality” [1], [2], i.e., the number of rules increases exponentially as the number of variables increases. Recently, some off-line paradigms have been proposed for structure identification; see [4]–[8] and [10]–[15] for details. Furthermore, several on-line dynamic schemes have been developed to determine the space partition [16]–[18]. In [8] and [18], the space partition is essentially a cluster method, which is widely used for classification problems. We feel that this method is theoretically not suitable for function approximation because two samples close to each other in the input space do not necessarily have similar outputs. The proposed GD-FNN, similar to [16] and [17], need not partition the input space a priori. It partitions the input–output space on-line dynamically according to both performance and accommodation boundary.

B. Ellipsoidal Regions and Estimation of Widths

Besides the locations of fuzzy rules, it has been shown that the region of fuzzy rules also plays an important role for a fuzzy system [25], and several other researchers have focused on developing fuzzy systems with ellipsoidal regions [18], [26], [27]. In [18], the initial widths are selected randomly because they will be further tuned by the BP algorithm. In [26], the widths are first determined by unsupervised competitive learning and subsequently modified by the BP method. As we have shown in (6) and (7), the widths could also be directly computed via covariance matrix, which is adopted in [27]. However, our results show that the widths chosen by the covariance matrix are too small for many rules to be generated. The same observation was also demonstrated in [27].

C. Relations to Other Algorithms

The OLS method was often used in fuzzy systems and neural networks as selection of significant neurons [20] or fuzzy basis functions [5] because of its fast learning speed. Actually, this idea is also adopted in our algorithm to judge which rules should be accepted and which widths could be modified. The results in Tables II and III show that this method is not as good as the GD-FNN, especially when we consider its generalization. The main reason is that the parameters of Gaussian functions are determined independently (usually selected as predefined values
and fixed afterward). Actually, the parameters, especially the widths, have great influence on generalization. A large width of a Gaussian function indicates that the function is not sensitive to the corresponding input variable and can be regarded as global approximation. On the contrary, the Gaussian function with small width is usually viewed as local approximation.

Table VII shows the receptive fields in the GD-FNN and D-FNN in Example 2, and the membership functions resulted from the D-FNN are depicted in Fig. 13.

We see from Table III that the GD-FNN achieves better performance than the D-FNN even if they have the same rules and same parameters. The training error will be decreased significantly if two more rules are generated. From Table VII and Fig. 13, we also find that two of the membership functions in the D-FNN have large widths ($\sigma = 12.2$ and $\sigma = 9.35$) so that their outputs in operate interval are always near 1, which are opaque for users to understand. On the other hand, similar membership functions generated by the D-FNN may occur [see Fig. 13(a) and (b)].

Another important issue related to the D-FNN is that the predefined parameters are difficult to select a priori for different systems. One method to alleviate this is to normalize the training data into a uniform map $[-1, 1]$ as follows:

\[
X^k = \frac{X^k}{\max\left(\left\{X^k_{\min}, X^k_{\max}\right\}\right)}, \quad T^k = \frac{T^k}{\max\left(\left\{T^k_{\min}, T^k_{\max}\right\}\right)} \tag{52}
\]

Instead of normalization, the GD-FNN is based on fuzzy $\varepsilon$-completeness, and a novel on-line parameter allocation mechanism is proposed to alleviate the random choice of initialization. It is very easy to configure a fuzzy system with good performance.

### VI. Conclusion

In this paper, a general learning paradigm is presented to automatically extract fuzzy rules from training data for the problem of function approximation. The main contributions of this paper are as follows.

1) On-line self-organizing learning is developed so that structure and parameters identification are done automatically and simultaneously without partitioning the input space and selecting initial parameters a priori.

![Fig. 13. Membership functions produced by the D-FNN of input variables (a) $u(t)$, (b) $y(t-1)$, and (c) $y(t)$ in RBF units.](image)
2) An on-line allocation method of EBF parameters is developed to avoid random choice based on fuzzy $\varepsilon$-completeness. 
3) Sensitivities of both input variables and fuzzy rules to the system performance are analyzed, and the widths of input variables in each rule can be self-adaptive according to its contribution to the system performance.

This proposed algorithm has fast learning speed instead of iterative learning. The effectiveness of the proposed approach is demonstrated in static function approximation, nonlinear dynamic system identification, time-varying system, and complex system with uncertainty. Simulation results show that compact and high-performance fuzzy rule-base can be created by using the proposed GD-FNN. Comprehensive comparisons with other latest approaches show that the proposed approach is superior in terms of learning efficiency and performance.

The major problem in this method is to determine several predefined parameters heuristically. Although these parameters offer flexible tradeoff between complexity and performance, inappropriate choice of these parameters may result in the problem of oscillation. The robustness of the algorithm and estimation of performance boundary are currently under investigation.

ACKNOWLEDGMENT

The authors would like to thank X. H. Meng for her help and discussions in time-varying drug delivery control system. The authors would also like to thank the anonymous reviewers for their valuable comments.

REFERENCES

Yang Gao received the B.Eng. degree (first-class honors) in electrical and electronic engineering from the Nanyang Technological University, Singapore, in 2000. She is currently working toward the Ph.D. degree at the same university.

Her research interests include robotics systems, adaptive control, robust control, and intelligent control using fuzzy neural techniques.