A Self-Evolving Interval Type-2 Fuzzy Neural Network With Online Structure and Parameter Learning

Chia-Feng Juang, *Senior Member, IEEE*, and Yu-Wei Tsao

***Abstract*—This paper proposes a self-evolving interval type-2 fuzzy neural network (SEIT2FNN) with online structure and parameter learning. The antecedent parts in each fuzzy rule of the SEIT2FNN are interval type-2 fuzzy sets and the fuzzy rules are of the Takagi–Sugeno–Kang (TSK) type. The initial rule base in the SEIT2FNN is empty, and the online clustering method is proposed to generate fuzzy rules that flexibly partition the input space. To avoid generating highly overlapping fuzzy sets in each input variable, an efficient fuzzy set reduction method is also proposed. This method independently determines whether a corresponding fuzzy set should be generated in each input variable when a new fuzzy rule is generated. For parameter learning, the consequent part parameters are tuned by the rule-ordered Kalman filter algorithm for high-accuracy learning performance. Detailed learning equations on applying the rule-ordered Kalman filter algorithm to the SEIT2FNN consequent part learning, with rules being generated online, are derived. The antecedent part parameters are learned by gradient descent algorithms. The SEIT2FNN is applied to simulations on nonlinear plant modeling, adaptive noise cancellation, and chaotic signal prediction. Comparisons with other type-1 and type-2 fuzzy systems in these examples verify the performance of the SEIT2FNN.**

***Index Terms*—Evolving system, fuzzy neural networks (FNNs), online fuzzy clustering, structure learning, type-2 fuzzy systems.**

I. Introduction

FUZZY logic systems (FLSs) have been successfully used in a variety of areas. However, most of these studies focused on type-1 FLSs. In recent years, studies on type-2 FLSs have drawn much attention [1], [2] and a complete theory of type-2 FLSs has been developed in [3]. Type-2 FLSs are extensions of type-1 FLSs, where the membership value of a type-2 fuzzy set is a type-1 fuzzy number. Type-2 FLSs appear to be a more promising method than their type-1 counterparts in handling problems with uncertainties such as noisy data and different word meanings. That is, type-2 fuzzy sets allow researchers to model and minimize the effects of uncertainties in rule-based systems. Successful applications of type-2 FLSs include areas of signal processing [4], [5], control [6]–[8], and medical applications [9], [10].

Fuzzy rule derivation is often difficult and time-consuming, and requires expert knowledge. This creates a common bottleneck in the fuzzy system design. To overcome this disadvantage, researchers have proposed automatic fuzzy system optimization methods. Some design methods were proposed via selecting significant fuzzy rules from an initial large rule base, such as clustering [11], singular value QR decomposition [12], and orthogonal least squares [13]. Apart from the aforementioned methods, most automatic fuzzy system optimization methods used neural learning algorithms [14]–[19]. These learning systems are usually called fuzzy neural networks (FNNs). The author in [14] proposed an adaptive-network-based fuzzy inference system (ANFIS). The ANFIS structure is fixed, and all ANFIS parameters are tuned by a gradient descent algorithm. Many FNNs with structure learning had been proposed. The studies in [15] and [16] used a subtractive clustering method for rule merging and reduction. The authors in [17] proposed a heuristic self-organizing network for structure learning. That method is based on k-means, and the number of clusters has to be decided in advance, which is unsuitable for online learning. The dynamic evolving neural fuzzy inference system (DENFIS) proposed in [18] used a general Euclidean distance as an online rule generation criterion. This criterion does not take the width of each fuzzy set into consideration. The evolving Takagi–Sugeno (ETS) model in [19] used the potential of new data instead of distance to a rule center for rule generation criterion. The potential criterion was shown to be more reasonable than the distance criterion; however, potential calculation was computationally expensive. In addition to fuzzy modeling accuracy, some works on fuzzy systems design with the consideration of system interpretability have been proposed [20]–[23]. Most FNNs are proposed for the type-1 FLSs design. Type-2 fuzzy rules are more complex than type-1 fuzzy rules because of their use of type-2 fuzzy sets in antecedent or consequent parts. Therefore, most type-2 FNN research is only concerned with interval type-2 fuzzy systems [24]–[28].

The theory of interval type-2 fuzzy systems was studied in [24]. In [25], the authors proposed parameter learning of an interval type-2 fuzzy system using Gaussian primary membership functions (MFs) with uncertain standard deviation as a type-2 fuzzy set. Studies [26]–[28] proposed parameter learning algorithms of interval type-2 fuzzy systems using Gaussian primary MFs with uncertain means as type-2 fuzzy sets. Optimal training for an interval type-2 FNN using a gradient descent algorithm

Manuscript received April 4, 2007; revised July 15, 2007 and October 19, 2007; accepted October 26, 2007. First published May 23, 2008; current version published December 19, 2008. This work was supported in part by the National Science Council, R.O.C., under Grant NSC-96-2628-E-005-087 and in part by the Ministry of Education, Taiwan, R.O.C., under the ATU plan.

C.-F. Juang is with the Department of Electrical Engineering, National Chung-Hsing University, Taichung 402, Taiwan, R.O.C. (e-mail: cfjuang@dragon.nchu.edu.tw).

Y.-W. Tsao is with the Industrial Technology Research Institute, Hsinchu 300, Taiwan, R.O.C.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2008.925907

was proposed in [26], and further corrected in a subsequent work [28]. However, this study's parameter learning equations still did not consider the fuzzy rule reordering problem during the computation of interval left and right end points of output (as discussed in [27]). Complete and detailed parameter-learning equations that considered the fuzzy rule reordering problem using a gradient descent algorithm were proposed in [27]. In addition to gradient descent algorithm tuning, the use of the recursive least square (RLS) method was also proposed in [29] for consequent part parameter learning in an interval type-2 fuzzy system. However, that study did not derive the learning equations or discuss the fuzzy rule-ordering problem.

In the aforementioned type-2 FNNs, only parameters are learned, and the structures are all fixed and must be assigned in advance. Fixed structures may not handle time-varying systems or systems with operating points changing with time. Therefore, the purpose of this paper is to develop a type-2 FNN with both structure and parameter evolution ability. The proposed FNN is called a self-evolving interval type-2 fuzzy neural network (SEIT2FNN). Initially, there are no fuzzy rules in an SEIT2FNN. All of the rules are generated online by proposed structure learning that not only helps automate rule generation, but also locates good initial rule positions for subsequent parameter learning. In contrast to the aforementioned rule-growing type-1 FNNs, an SEIT2FNN directly uses the firing strength as rule generation criterion, which is simple and requires no additional criterion computation. In addition, SEIT2FNN avoids the generation of highly overlapping fuzzy sets. This phenomenon was not considered in the aforementioned type-1 FNNs. The SEIT2FNN's structure evolution ability makes it more suitable for handling time-varying systems than type-2 fuzzy systems that adjust their parameters based on pretrained and fixed structures. The consequent parameters in the SEIT2FNN are learned by a rule-ordered Kalman filter algorithm, which derives detailed learning equations considering the rule reordering problem. The antecedent part parameters are learned by gradient descent learning algorithms. Several simulations are conducted to verify SEIT2FNN performance. These simulations also compare other type-1 and type-2 systems.

The rest of this paper is organized as follows. Section II introduces the SEIT2FNN structure. Section III introduces structure and parameter learning in an SEIT2FNN. Section IV simulates five examples, which include system identification, adaptive noise cancellation, and chaotic sequence prediction. Finally, Section V draws conclusions.

II. SEIT2FNN Structure

This section introduces the structure of an SEIT2FNN. Fig. 1 shows the proposed network structure, which has a total of six layers. This six-layered network realizes an interval type-2 fuzzy system whose consequent part is a linear combination of input variables, i.e., Takagi–Sugeno–Kang (TSK) type. Each SEIT2FNN rule has the following form:

$$\text{Rule } i\text{: IF } x_1 \text{ is } \tilde{A}_1^i \text{ AND} \ldots \text{AND } x_n \text{ is } \tilde{A}_n^i$$
$$\text{THEN } y \text{ is } \tilde{a}_0^i + \sum_{j=1}^{n} \tilde{a}_j^i x_j, \qquad i = 1, \ldots, M \quad (1)$$

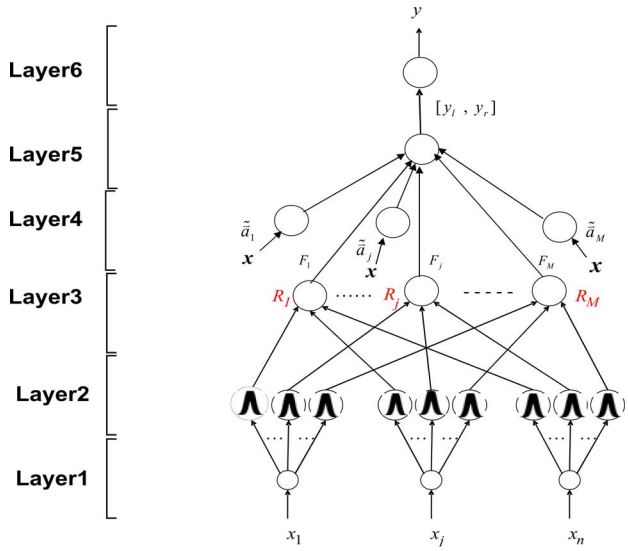

Fig. 1. Structure of the SEIT2FNN.

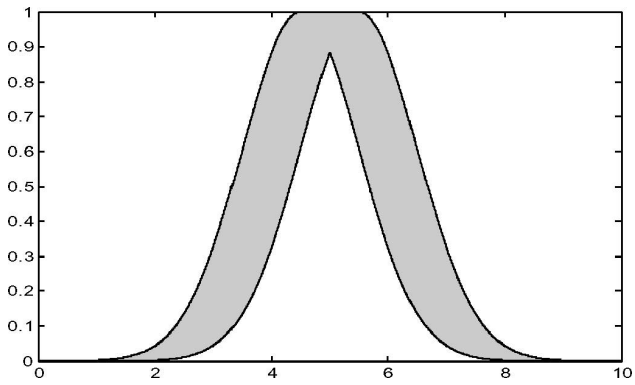

Fig. 2. Interval type-2 fuzzy set with an uncertain mean.

where $\tilde{A}_j^i$'s, $j = 1, \ldots, n$, are interval type-2 fuzzy sets, M is the number of rules, $\tilde{a}_j^i$'s, $j = 0, \ldots, n$, are interval sets, and

$$\tilde{a}_j^i = [c_j^i - s_j^i, c_j^i + s_j^i], \qquad j = 0, \ldots, n. \quad (2)$$

Detailed mathematical functions of each layer are introduced as follows.

1) *Layer 1 (Input Layer):* The inputs are crisp values. For input range normalization, each node in this layer scales inputs x_i, $i = 1, \ldots, n$, to within the range $[-1,\ 1]$. Note that there are no weights to be adjusted in this layer.
2) *Layer 2 (Fuzzification Layer):* This layer performs the fuzzification operation. Each node in this layer defines an interval type-2 MF. For the *i*th fuzzy set $\tilde{A}_j^i$ in input variable x_j, a Gaussian primary MF, having a fixed standard deviation σ and an uncertain mean that takes on values in $[m_1,\ m_2]$, is used (see Fig. 2), i.e.,

$$\mu_{\tilde{A}_j^i} = \exp\left[-\frac{1}{2}\left(\frac{x_j - m_j^i}{\sigma_j^i}\right)^2\right] \equiv N(m_j^i, \sigma_j^i; x_j),$$
$$m_j^i \in \left[m_{j1}^i, m_{j2}^i\right]. \quad (3)$$

The footprint of uncertainty (FoU) of this MF can be represented as a bounded interval in terms of the upper MF, $\overline{\mu}_{\tilde{A}_j^i}$, and lower MF, $\underline{\mu}_{\tilde{A}_j^i}$, where

$$\overline{\mu}_{\tilde{A}_j^i}(x_j) = \begin{cases} N(m_{j1}^i, \sigma_j^i; x_j), & x_j < m_{j1}^i \\ 1, & m_{j1}^i \le x_j \le m_{j2}^i \\ N(m_{j2}^i, \sigma_j^i; x_j), & x_j > m_{j2}^i \end{cases} \tag{4}$$

and

$$\underline{\mu}_{\tilde{A}_j^i}(x_j) = \begin{cases} N(m_{j2}^i, \sigma_j^i; x_j), & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ N(m_{j1}^i, \sigma_j^i; x_j), & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2}. \end{cases} \tag{5}$$

That is, the output of each node can be represented as an interval $[\underline{\mu}_{\tilde{A}_j^i}, \overline{\mu}_{\tilde{A}_j^i}]$.

3) *Layer 3 (Firing Layer):* Each node in this layer is a rule node, and performs the fuzzy meet operation [24] using an algebraic product operation. The output of a rule node is the firing strength F^i, which is an interval type-1 fuzzy set. The firing strength is computed as follows [24]:

$$F^i = [\underline{f}^i, \overline{f}^i] \tag{6}$$

where

$$\overline{f}_i = \prod_{j=1}^{n} \overline{\mu}_{\tilde{A}_j^i} \quad \text{and} \quad \underline{f}_i = \prod_{j=1}^{n} \underline{\mu}_{\tilde{A}_j^i}. \tag{7}$$

4) *Layer 4 (Consequent Layer):* Each node in this layer is called a consequent node. Each rule node in Layer 3 has its own corresponding consequent node in Layer 4. The output of each node is an interval type-1 fuzzy set, denoted by $[w_l^i, w_r^i]$. According to (1) and (2), the node output is

$$[w_l^i, w_r^i] = [c_0^i - s_0^i, c_0^i + s_0^i] + \sum_{j=1}^{n} [c_j^i - s_j^i, c_j^i + s_j^i] x_j. \tag{8}$$

That is,

$$w_l^i = \sum_{j=0}^{n} c_j^i x_j - \sum_{j=0}^{n} |x_j|\, s_j^i \tag{9}$$

where $x_0 \triangleq 1$, and

$$w_r^i = \sum_{j=0}^{n} c_j^i x_j + \sum_{j=0}^{n} |x_j|\, s_j^i. \tag{10}$$

5) *Layer 5 (Output Processing Layer):* The extended output is an interval type-1 set $[y_l, y_r]$, where indexes l and r mean left and right limits, respectively. Each node in this layer computes this interval output. The outputs y_l and y_r can be computed using the Karnik–Mendel iterative procedure [3]. In that procedure, the consequent parameters are reordered in ascending order. Let $\boldsymbol{w_l} = (w_l^1, \ldots, w_l^M)$ and $\boldsymbol{w_r} = (w_r^1, \ldots, w_r^M)$ denote the original rule-ordered consequent values, and let $\boldsymbol{y_l} = (y_l^1, \ldots, y_l^M)$ and $\boldsymbol{y_r} = (y_r^1, \ldots, y_r^M)$ denote the reordered sequence, where $y_l^1 \le y_l^2 \le \ldots \le y_l^M$ and $y_r^1 \le y_r^2 \le \ldots \le y_r^M$. According to [27], the relationship between $\boldsymbol{w_l}$, $\boldsymbol{w_r}$, $\boldsymbol{y_l}$, and $\boldsymbol{y_r}$ is

$$\boldsymbol{y}_l = Q_l \boldsymbol{w_l} \quad \text{and} \quad \boldsymbol{y_r} = Q_r \boldsymbol{w_r} \tag{11}$$

where Q_l and Q_r are $M \times M$ permutation matrices. These two permutation matrices used elementary vectors (i.e., vectors, all of whose elements are zero except one element, whose value is equal to one) as columns, and these vectors are arranged (permuted) so as to move elements in $\boldsymbol{w_l}$ and $\boldsymbol{w_r}$ to new locations in ascending order in the transformed vectors $\boldsymbol{y_l}$ and $\boldsymbol{y_r}$, respectively. Reorder the f^i's accordingly and call them g^i's. The outputs y_l and y_r can be computed as follows:

$$y_l = \frac{\sum_{i=1}^{L} \overline{g}^i y_l^i + \sum_{i=L+1}^{M} \underline{g}^i y_l^i}{\sum_{i=1}^{L} \overline{g}^i + \sum_{i=L+1}^{M} \underline{g}^i} \tag{12}$$

and

$$y_r = \frac{\sum_{i=1}^{R} \underline{g}^i y_r^i + \sum_{i=R+1}^{M} \overline{g}^i y_r^i}{\sum_{i=1}^{R} \underline{g}^i + \sum_{i=R+1}^{M} \overline{g}^i}. \tag{13}$$

6) *Layer 6 (Output Layer):* Each output node corresponds to one output linguistic variable. Nodes in this layer compute the output linguistic variable y using a defuzzification operation. Because the output of Layer 5 is an interval set, nodes in Layer 6 defuzzify it by computing the average of y_l and y_r. Hence, the defuzzified output is

$$y = \frac{y_l + y_r}{2}. \tag{14}$$

If the output variable is too large or too small in contrast to the range $[-1, 1]$, then it would be more efficient for training if the output value is normalized to within the range $[-1, 1]$. During test, the output variable should be denormalized to the original range to get the actual output.

III. SEIT2FNN Learning

The SEIT2FNN simultaneously uses two types of learning to evolve: structure and parameter learning. There are no rules in the SEIT2FNN initially. The rules are created online as learning proceeds upon receiving the training data. After a rule is generated, its free parameters are tuned by subsequent parameter learning. The following sections introduce detailed structure and parameter-learning algorithms.

A. Structure Learning

The first task in structure learning is to determine when to generate a new rule. The way the input space is partitioned determines the number of rules extracted from the training data, as well as the number of fuzzy sets in the universe of discourse for each input variable. Geometrically, a rule corresponds to a cluster in the input space, and a rule firing strength can be regarded as the degree to which input data belong to the cluster. Based on this concept, a previous study [30] used the rule

firing strength as a criterion for type-1 fuzzy rule generation. This idea is extended to type-2 fuzzy rule generation criteria in an SEIT2FNN. Since the firing strength in the SEIT2FNN is an interval [see (6)], the center of the interval is computed as

$$f_c^i = \frac{1}{2}(\overline{f}^i + \underline{f}_i). \tag{15}$$

The firing strength center then serves as a rule generation criterion. That is, for each piece of incoming data $\vec{x} = (x_1, \ldots, x_n)$, find

$$I = \arg \max_{1 \le i \le M(t)} f_c^i(\vec{x}) \tag{16}$$

where $M(t)$ is the number of existing rules at time t. If $f_c^I(\vec{x}) \le \phi_{\text{th}}$, then a new rule is generated, where $\phi_{\text{th}} \in (0,\ 1)$ is a prespecified threshold. The threshold decides the number of input clusters generated in an SEIT2FNN. A smaller ϕ_{th} value generates a smaller number of rules. Once a new rule is generated, there is a problem in determining if the corresponding fuzzy set in each input variable should be generated. One simple solution is to generate a new fuzzy set in each input variable for a new fuzzy rule. However, this method usually generates highly overlapping fuzzy sets in each input variable. An alternative method is proposed in [30], where identical widths are initially set to the n possible new fuzzy sets corresponding to the new rule. After initial width assignment, fuzzy similarity measurements between two type-1 fuzzy sets are computed and serve as criteria for new fuzzy set generation. However, this approach does not consider the one-dimensional (1-D) geometric relationship between the input data and existing fuzzy sets in each input variable. In addition, computing the degree of similarity between two fuzzy sets is very complex in [30]. The similarity between two Gaussian fuzzy sets A and B is computed using $|A \cap B|/|A \cup B|$ in [30], where $|A \cap B|$ and $|A \cup B|$ denote the intersection and union areas between two Gaussian fuzzy sets, respectively. Computation of these areas is very complex even if the Gaussian MFs are approximated by triangular MFs [30]. Though the literature on similarity measures for type-1 fuzzy sets is quite extensive and some are simple in computation [31], only about five similarity measures for interval type-2 fuzzy sets have appeared to date, as studied in [32]. The SEIT2FNN uses interval type-2 fuzzy sets, and similarity measure computation using existing methods is complex. The SEIT2FNN addresses these problems by proposing a novel and efficient fuzzy set reduction method, as introduced later.

The idea behind (16) is reapplied to 1-D space in an SEIT2FNN, where the membership value $[\underline{\mu}_{\tilde{A}_j^i}, \overline{\mu}_{\tilde{A}_j^i}]$ in layer 2 [see (4) and (5)] is used as a fuzzy set generation criterion. Since this value is an interval, its center $\mu^c_{\tilde{A}_j^i}$ is computed as

$$\mu^c_{\tilde{A}_j^i} = \frac{1}{2}(\underline{\mu}_{\tilde{A}_j^i} + \overline{\mu}_{\tilde{A}_j^i}). \tag{17}$$

For each newly generated rule, compute

$$I_j = \arg \max_{1 \le i \le k_j(t)} \mu^c_{\tilde{A}_j^i}, \qquad j = 1, \ldots, n \tag{18}$$

where $k_j(t)$ is the number of fuzzy sets in input variable j. If $\mu_{\tilde{A}_j^{I_j}} > \rho$, where $\rho \in [0, 1]$, is a prespecified threshold, then use the existing fuzzy set $\tilde{A}_j^{I_j}$ as the antecedent part of the new rule in input variable j. Otherwise, generate a new fuzzy set in input variable j and set $k_j(t+1) = k_j(t) + 1$. Parameter ρ determines the number of fuzzy sets in each input variable. A smaller value of ρ generates a smaller number of fuzzy sets. If ρ is set to one, then the number of fuzzy sets in each input variable is equal to the number of rules. On the contrary, if ρ is set to zero, then only one fuzzy set is generated in each input variable. Parameter ρ is recommended to be selected from the range [0.4, 0.9] if the fuzzy set reduction method is used.

Once a new fuzzy set is generated, it should be assigned with an initial shape. From (18), the interval type-2 fuzzy set I_j possesses the maximum degree to which x_j belongs. In an SEIT2FNN, the initial width of a new fuzzy set in input variable j depends on the distance between input x_j and the uncertain mean average of this fuzzy set. According to the notation in (3), the initial uncertain mean m_j^i and standard deviation σ_j^i for the $k_j(t+1)$th interval type-2 fuzzy set in input variable x_j are

$$m_j^i \in [x_j - 0.1, x_j + 0.1] \tag{19}$$

$$\sigma_{k_j(t+1)j} = \beta \left| x_j - \frac{1}{2}\left(m_{j1}^{I_j} + m_{j2}^{I_j}\right) \right| \tag{20}$$

where $\beta > 0$ determines the overlap degree between two fuzzy sets. The phenomenon of high overlapping between fuzzy sets occurs when β is too large, and there is almost no overlapping if β is too small. This paper sets β at 0.5 so that the width of the new fuzzy set is half the distance between the average centers of the new fuzzy set and fuzzy set I_j, and a suitable overlapping between these two fuzzy sets is generated.

In addition to assigning the initial antecedent part parameters, the initial consequent part parameters should also be determined for a new generated rule. The initial consequent parameters are set to

$$[c_0^i - s_0^i, c_0^i + s_0^i] = [y_d - 0.1, y_d + 0.1], \qquad i = 1, \ldots, M \tag{21}$$

where y_d is the desired output for input $\vec{x}$. The initial parameter s_0^i determines the initial output interval range. If this initial parameter is too small, e.g., 0.01, then the output interval would be very close to a point. On the contrary, if this parameter is too large, e.g., 0.5, then the output interval would cover the entire output range. Thus, (21) sets s_0^i at the appropriate value of 0.1. Initial parameters c_j^i, $j = 1, \ldots, n$, are set at very small values, and initial s_j^i, $j = 1, \ldots, n$, are the same as s_0^i. That is,

$$c_j^i = 0.01, \qquad s_j^i = s_0^i = 0.1, \qquad j = 1, \ldots, n, \quad \text{and} \quad i = 1, \ldots, M. \tag{22}$$

Repeating the earlier process for every incoming piece of training data generates new rules, one after another, until the complete SEIT2FNN is finally constructed. The entire learning algorithm for generating new fuzzy rules as well as fuzzy sets in each input variable is as follows.

Algorithm (structure learning)
IF $\vec{x}$ *is the first incoming data THEN do*
{ Generate a new fuzzy rule and assign initial width and center of each antecedent fuzzy set by
$m_j^1 \in [x_j - 0.1,\ x_j + 0.1]$ and σ_{1j}=0.4, $j = 1,...,n$
}
ELSE for each newly incoming data $\vec{x}$ *do*
{Compute (16)
IF $f_c^I(\vec{x}) \le \phi_{th}$
{ $M(t+1) = M(t)+1$,
For $j = 1,\ldots,n$
{ Compute (18).
IF $\mu_{\tilde{A}_j^{I_j}} > \rho$
THEN use the fuzzy set $\tilde{A}_j^{I_j}$ *as the antecedent part of rule* $M(t+1)$
ELSE
Generate a new fuzzy set, $k_j(t+1) = k_j(t)+1$,
Set initial width and mean of the new fuzzy set by (21) and (22).
} } }

In the previous algorithm, the initial width σ_{1j} of the first rule is set at 0.4. Since the inputs are scaled within the range $[-1, 1]$ in layer 1 of SEIT2FNN, this initial width is set intuitively according to the input range so that the first rule geometrically covers a suitable region in the input space. Hence, similar values can be assigned. If σ_{1j} is assigned too large a value, e.g., $\sigma_{1j} = 0.9$, then the initial cluster would cover almost all the input space, which is obviously unsuitable. On the contrary, if σ_{1j} is assigned too small a value, e.g., $\sigma_{1j} = 0.01$, then this initial rule would have almost no effect in the whole SEIT2FNN.

Structure learning proceeds for each piece of incoming data despite it being a new sample during online training or an old sample from repeated off-line training. The aforementioned learning algorithm shows that SEIT2FNN automatically stops the generation of rules if $f_c^I(\vec{x}) > \phi_{\text{th}}$ for all input data $\vec{x}$, indicating that all input data have been suitably covered by the existing fuzzy rules geometrically. Parameters of each evolved rules are modified by the parameter-learning algorithm introduced in the next section.

B. Parameter Learning

The parameter-learning phase occurs concurrently with the structure-learning phase. For each piece of incoming data, all free SEIT2FNN parameters are tuned, whether the rules are newly generated or originally existent. To clarify, consider the single-output case, where the objective is to minimize the error function

$$E = \frac{1}{2}[y(t) - y_d(t)]^2. \tag{23}$$

Here, $y(t)$ and $y_d(t)$ denote real and desired outputs, respectively. The Karnik–Mendel iterative procedure for computing y_l and y_r in (12) and (13) has the premise that w_l^i and w_r^i are rearranged in ascending order. During the parameter learning process, the values of w_l^i and w_r^i change, and their orders and corresponding rule orders in computing (12) and (13) should change accordingly. To update the parameters, it is necessary to know exactly where specific antecedent and consequent parameters are located, and this is very difficult to ascertain when y_l and y_r are not in a rule-ordered format. This problem was addressed in [27]. However, studies [26], [28], and [29] do not clearly address this problem, and direct use of the derived learning algorithms in those studies without considering the rule-ordering problem may cause errors. The SEIT2FNN approach considers this problem in its derived rule-ordered Kalman filtering algorithm for consequent parameter learning. Let $\underline{\boldsymbol{f}} = (\underline{f}^1, \underline{f}^2, \ldots, \underline{f}^M)^T$ and $\overline{\boldsymbol{f}} = (\overline{f}^1, \overline{f}^2, \ldots, \overline{f}^M)^T$ where the firing strengths are expressed according to the original rule order. According to [27], (12) can be reexpressed in the following rule-ordered form:

$$y_l = \frac{\overline{\boldsymbol{f}}^T Q_l^T E_1^T E_1 Q_l \boldsymbol{w_l} + \underline{\boldsymbol{f}}^T Q_l^T E_2^T E_2 Q_l \boldsymbol{w_l}}{\sum_{i=1}^{L} (Q_l \overline{\boldsymbol{f}})_i + \sum_{i=L+1}^{M} (Q_l \underline{\boldsymbol{f}})_i} \tag{24}$$

where

$$E_1 = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_L, \boldsymbol{0}, \ldots, \boldsymbol{0}) \in \Re^{L\times M},$$
$$E_2 = (\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_{M-L}) \in \Re^{(M-L)\times M} \tag{25}$$

and where $\boldsymbol{e}_i \in \Re^{L\times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{M-L}$ are elementary vectors. Similarly, (13) can be reexpressed in the following rule-ordered form:

$$y_r = \frac{\underline{\boldsymbol{f}}^T Q_r^T E_3^T E_3 Q_r \boldsymbol{w_r} + \overline{\boldsymbol{f}}^T Q_r^T E_4^T E_4 Q_r \boldsymbol{w_r}}{\sum_{i=1}^{R} (Q_r \underline{\boldsymbol{f}})_i + \sum_{i=R+1}^{M} (Q_r \overline{\boldsymbol{f}})_i} \tag{26}$$

where

$$E_3 = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_R, \boldsymbol{0}, \ldots, \boldsymbol{0}) \in \Re^{R\times M},$$
$$E_4 = (\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_{M-R}) \in \Re^{(M-R)\times M} \tag{27}$$

and where $\boldsymbol{e}_i \in \Re^{R\times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{(M-R)\times 1}$ are elementary vectors. Consequent parameters are tuned in [27] by a gradient descent algorithm. In an SEIT2FNN, a rule-ordered Kalman filtering algorithm tunes consequent parameters. Equations (24) and (26) can be reexpressed as

$$y_l = \boldsymbol{\phi}_l^{\boldsymbol{T}} \boldsymbol{w}_l, \qquad \boldsymbol{\phi}_l \in \Re^{M\times 1} \tag{28}$$

and

$$y_r = \boldsymbol{\phi}_r^{\boldsymbol{T}} \boldsymbol{w}_r, \qquad \boldsymbol{\phi}_r \in \Re^{M\times 1} \tag{29}$$

respectively. Thus, the output y in (14) can be reexpressed as

$$y = \frac{1}{2}(y_l + y_r) = \frac{1}{2}\left(\boldsymbol{\phi}_l^{\boldsymbol{T}}\boldsymbol{w}_l + \boldsymbol{\phi}_r^{\boldsymbol{T}}\boldsymbol{w}_r\right) = \begin{bmatrix}\overline{\boldsymbol{\phi}}_l^{\boldsymbol{T}} & \overline{\boldsymbol{\phi}}_r^{T}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_l \\ \boldsymbol{w}_r\end{bmatrix}$$

$$= \begin{bmatrix}\overline{\phi}_{l1}\cdots\overline{\phi}_{lM(t)} & \overline{\phi}_{r1}\cdots\overline{\phi}_{rM(t)}\end{bmatrix}\begin{bmatrix} w_{l1} \\ \vdots \\ w_{lM(t)} \\ w_{r1} \\ \vdots \\ w_{rM(t)}\end{bmatrix} \tag{30}$$

where $\overline{\boldsymbol{\phi}}_l^{T} = 0.5\boldsymbol{\phi}_l^{T}$ and $\overline{\boldsymbol{\phi}}_r^{\boldsymbol{T}} = 0.5\boldsymbol{\phi}_r^{\boldsymbol{T}}$. According to (9) and (10), (30) can be further expressed as follows:

$$y = \begin{bmatrix}\overline{\boldsymbol{\phi}}_l^{\boldsymbol{T}} & \overline{\boldsymbol{\phi}}_r^{\boldsymbol{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{w}_l \\ \boldsymbol{w}_r\end{bmatrix}$$

$$= \begin{bmatrix}\overline{\phi}_{l1}\cdots\overline{\phi}_{lM(t)} & \overline{\phi}_{r1}\cdots\overline{\phi}_{rM(t)}\end{bmatrix}\begin{bmatrix} \sum_{j=0}^{n} c_j^1 x_j - \sum_{j=0}^{n}|x_j|\, s_j^1 \\ \vdots \\ \sum_{j=0}^{n} c_j^M x_j - \sum_{j=0}^{n}|x_j|\, s_j^M \\ \sum_{j=0}^{n} c_j^1 x_j + \sum_{j=0}^{n}|x_j|\, s_j^1 \\ \vdots \\ \sum_{j=1}^{n} c_0^M x_j + \sum_{j=0}^{n}|x_j|\, s_j^M\end{bmatrix}. \tag{31}$$

Since SEIT2FNN rules are generated online, the dimension of $\boldsymbol{w}_l$ in (31) increases with time and the positions of c_j^i and s_j^i in a vector change accordingly. To maintain constant positions of c_j^i and s_j^i in a vector, vector components in (31) are rearranged in rule order in the proposed rule-ordered Kalman filtering algorithm. Let $\boldsymbol{w}_{\mathrm{TSK}} \in \Re^{2M(n+1)\times 1}$ denote all the consequent parameters c_j^i and s_j^i, $j = 0,\ldots,n$, $i = 1,\ldots,M$, i.e.,

$$\boldsymbol{w}_{\mathrm{TSK}} = [c_0^1 \ldots c_n^1 \quad s_0^1 \ldots s_n^1 \ldots\ldots c_0^M \ldots c_n^M \quad s_0^M \ldots s_n^M]^T \tag{32}$$

where the parameters are placed according to the rule order so that their positions remain constant as the rule number increases during structure learning. Equation (31) can then be expressed as

$$y = \begin{bmatrix}\overline{\phi}_{c1}x_0\cdots\overline{\phi}_{c1}x_n & -\overline{\phi}_{s1}|x_0|\cdots-\overline{\phi}_{s1}|x_n| \\ \cdots\overline{\phi}_{cM}x_0\cdots\overline{\phi}_{cM}x_n & -\overline{\phi}_{sM}|x_0|\cdots-\overline{\phi}_{sM}|x_n|\end{bmatrix}\boldsymbol{w}_{\mathrm{TSK}}$$

$$= \overline{\boldsymbol{\phi}}'^{T}_{\mathrm{TSK}}\boldsymbol{w}_{\mathrm{TSK}} \tag{33}$$

where $\overline{\phi}_{cj} = \overline{\phi}_{lj} + \overline{\phi}_{rj}$ and $\overline{\phi}_{sj} = \overline{\phi}_{rj} - \overline{\phi}_{lj}$, $j = 1,\ldots,M$. The consequent parameter vector $\boldsymbol{w}_{\mathrm{TSK}}$ is updated by executing the following rule-ordered Kalman filtering algorithm

$$\boldsymbol{w}_{\mathrm{TSK}}(t+1) = \boldsymbol{w}_{\mathrm{TSK}}(t) + S(t+1)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(t+1)(y^d(t+1) - \overline{\boldsymbol{\phi}}'^{T}_{\mathrm{TSK}}(t+1)\boldsymbol{w}_{\mathrm{TSK}}(t))$$

$$S(t+1) = \frac{1}{\lambda}\left[S(t) - \frac{S(t)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(\tau+1)\overline{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(t+1)S(t)}{\lambda + \overline{\boldsymbol{\phi}}'^{T}_{\mathrm{TSK}}(t+1)S(t)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}}\right] \tag{34}$$

where $0 < \lambda \leq 1$ is a forgetting factor ($\lambda = 0.9995$ in this paper). The dimensions of vectors $\boldsymbol{w}_{\mathrm{TSK}}$ and $\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}$, and matrix S increase when a new rule evolves. Let the dimensions of $\boldsymbol{w}_{\mathrm{TSK}}$ and S at time step t be $2M(n+1)$ and $2M(n+1)\times 2M(n+1)$, respectively. When a new rule evolves at time $t+1$, $\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(t+1)$ becomes

$$\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(t+1) = [\overline{\boldsymbol{\phi}}'^{T}_{\mathrm{TSK}}(t)\overline{\phi}_{cM+1}x_0\cdots\overline{\phi}_{cM+1}x_n - \overline{\phi}_{sM+1}|x_0|\cdots-\overline{\phi}_{sM+1}|x_n|]^T \in \Re^{2(M+1)(n+1)\times 1}. \tag{35}$$

SEIT2FNN augments $\boldsymbol{w}_{\mathrm{TSK}}(t)$ and $S(t)$ on the right-hand side of (34) as follows:

$$\boldsymbol{w}_{\mathrm{TSK}}(t) = \left[\boldsymbol{w}_{\mathrm{TSK}}^T(t) \;\; c_0^{M+1}\cdots c_n^{M+1} \;\; s_0^{M+1}\cdots s_n^{M+1}\right]^T \in \Re^{2(M+1)(n+1)\times 1} \tag{36}$$

and

$$S(t) = \text{block diag}[S(t) \quad qI] \in \Re^{2(M+1)(n+1)\times 2(M+1)(n+1)} \tag{37}$$

where $c_0^{M+1}\cdots c_n^{M+1}$ and $s_0^{M+1}\cdots s_n^{M+1}$ are initialized using (22) and q is a large positive constant. After augmentation, the dimensions of $\boldsymbol{w}_{\mathrm{TSK}}(t+1)$ and $S(t+1)$, computed using (34), become $2(M+1)(n+1)$ and $2(M+1)(n+1)\times 2(M+1)(n+1)$, respectively.

The theoretical convergence analysis of the Kalman filter for linear regression models or neural networks was studied in [33] and [34]. These studies analyze the Kalman filter with the fixed input dimension. In the SEIT2FNN, the input dimension to the Kalman filter varies with the fuzzy rule number. The theoretical convergence analysis of the Kalman filter with varying input dimensions is another research topic. This paper addresses this problem by resetting the matrix S to $q \cdot I$ after a period of learning time (ten learning iterations in this paper). This resetting operation keeps S bounded and helps avoid a divergence problem according to simulation results. Though this paper does not present theoretical analysis, simulation results in Section IV show that the learning of the SEIT2FNN empirically converges with good performance.

SEIT2FNN antecedent parameters are tuned by a gradient descent algorithm. That is

$$m_{j1}^i(t+1) = m_{j1}^i(t) - \eta\frac{\partial E}{\partial m_{j1}^i} \tag{38}$$

$$m_{j2}^i(t+1) = m_{j2}^i(t) - \eta\frac{\partial E}{\partial m_{j2}^i} \tag{39}$$

$$\sigma_j^i(t+1) = \sigma_j^i(t) - \eta\frac{\partial E}{\partial \sigma_j^i} \tag{40}$$

where η is a learning coefficient. The gradient descent algorithm as well as the ordered Kalman filter algorithm are performed with one iteration for each piece of incoming data despite it being an old sample from repeated offline learning or a new sample from online learning. Details of the learning algorithm can be found in [27], where the authors had explicitly derived gradient calculations considering the rules reordering problem. The

TABLE I
PERFORMANCE OF SEIT2FNN AND OTHER TYPE-1 AND TYPE-2 FUZZY SYSTEMS IN EXAMPLE 1

	Type 1		Type 2		
Methods	FALCON	SONFIN	T2FLS (Singleton)	T2FLS (TSK)	SEIT2FNN
Rule number	6	7	5	3	3
Parameter number	54	35	40	36	36
Iterations	60000	500	500	500	500
RMSE	0.02	0.008	0.034	0.0388	0.0062

gradient descent algorithm is used in SEIT2FNN for online learning consideration, though it may have a local minima problem. Global learning algorithms, like genetic algorithms [35] and particle swarm optimization [36], are less likely to be stuck in a local minimum. These algorithms are not used for SEIT2FNN evolution as they are unsuitable to online learning problems.

IV. SIMULATIONS

This section describes five SEIT2FNN simulation examples. These examples include system identification (Examples 1–3), adaptive noise cancellation (Example 4), and chaotic signal prediction (Example 5). These examples compare SEIT2FNN performance with other type-1 and type-2 FNNs.

A. Example 1—System Identification

This example uses the SEIT2FNN to identify a nonlinear system. The plant to be identified is guided by the difference equation [30], [37]

$$y_d(t+1) = \frac{y_d(t)}{1+y_d^2(t)} + u^3(t). \tag{41}$$

The training patterns are generated with $u(t) = \sin(2\pi t/100), t = 1, \ldots, 200$. The SEIT2FNN inputs are $y_d(t)$ and $u(t)$, and the desired output is $y_d(t+1)$. Performance is evaluated using the root mean square error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{200}\sum_{k=1}^{200}[y(t+1) - y_d(t+1)]^2} \tag{42}$$

where $y(t)$ is the SEIT2FNN output. The learning coefficient η is set at 0.05. The structure learning threshold ϕ_{th} determines the number of generated fuzzy rules. Three rules are generated when ϕ_{th} is set at 0.01. Since the number of fuzzy sets and the number of rules are small, the fuzzy set reduction operation in structure learning is not used, i.e., ρ is set to be 1. Training is performed for 500 iterations. Table I shows the performance of the SEIT2FNN. Table II shows the learned antecedent and consequent parameters in the three rules of the SEIT2FNN.

This example also tests SEIT2FNN performance with a greater number of rules, caused by assigning a larger value of ϕ_{th}. Table III shows the performance of the SEIT2FNN with a greater number of rules: ten rules are generated when $\phi_{\text{th}} = 0.15$ and $\rho = 1$. Fig. 3 shows the distributions of SEIT2FNN fuzzy sets using ten rules in inputs $y_d(k)$ and $u(k)$ without using the fuzzy set reduction operation. Here, there are also ten fuzzy sets in each input variable, and some are highly overlapped. To avoid this phenomenon, the proposed fuzzy set reduction operation with $\rho = 0.45$ is applied to this SEIT2FNN, and only five sets are generated in each input variable even though there are ten rules in all. Fig. 4 shows the distributions of the five fuzzy sets in each input where the highly overlapped phenomenon is avoided. Fig. 5 shows the corresponding identification results. Table III shows that the RMSE of the SEIT2FNN with fuzzy set reduction is similar to the RMSE without reduction; however, the number of fuzzy sets is greatly reduced.

The interval type-2 FLS presented in [3] and [27] is compared to the SEIT2FNN and denoted as T2FLS hereafter. The T2FLS has no structure learning, and the number of rules in T2FLS is assigned *a priori*. T2FLS parameters are tuned by the gradient descent algorithm in [27]. Tables I and III show T2FLS performance with singleton and TSK-type consequents, respectively. The results show that with the same number of rules and consequent type, the SEIT2FNN RMSE is much smaller than the T2FLS RMSE.

The performance of type-1 FNNs that were applied to the same identification problem is also compared. These type-1 FNNs include the fuzzy adaptive learning control network (FALCON) [37] and self-constructing neural fuzzy inference network (SONFIN) [30], both of which also have structure-learning abilities. Table I lists the performance of the FALCON and the SONFIN, indicating that the SEIT2FNN has a smaller RMSE and number of rules than these two type-1 FNNs. The results also show that SONFIN performance, which has structure-learning ability, is better than the T2FLS when a nearly identical number of parameters are used in both the networks.

Influences of η and ϕ_{th} on the SEIT2FNN performance are discussed. Table IV shows the learning results with different η values when $\phi_{\text{th}} = 0.05$ and the iteration number was 500. Six rules were generated after learning. The results show that smaller errors were obtained as η increased from 0.01 to 0.15. The performance when $\eta = 0.15$ was better than that when $\eta = 0.2$, but was worse than that when $\eta = 0.3$. When η was larger than 0.15, it was observed that the learning error curve oscillates and may produce a larger identification error. According to this observation, η was conservatively set to be smaller than 0.05 in the paper simulation examples to avoid unstable tuning results, though a larger η may achieve better performance.

Table V shows the learning results with different ϕ_{th} values when η was set at 0.05 and iteration number was 500. The results show that a higher number of ϕ_{th} generates a higher number of rules and generally improves the SEIT2FNN performance. However, when the rule number is too large, then the performance improvement saturates. One major reason is that the iteration number in Table V is fixed, and a higher iteration number is usually required to optimally tune the parameters in a larger network. To avoid the generation of a high rule number, examples in this paper set ϕ_{th} within the range [0.01, 0.15] when no fuzzy set reduction operation is used.

TABLE II
Learned Parameters in SEIT2FNN With Three Rules in Example 1

	Rule 1	Rule 2	Rule 3
$(m_{11}^i, m_{21}^i, \sigma_1^i)$	(-0.01698, -0.01648, 0.87812)	(0.68106 0.84087, 0.89093)	(-0.67939, -0.67914 0.84141)
$(m_{12}^i, m_{22}^i, \sigma_2^i)$	(-0.06955, 0.01729, 0.75122)	(0.67238 0.67240, 0.91077)	(-0.68762 -0.55597, 0.81405)
$(c_1^i,\ c_2^i,\ c_3^i)$	(0.00157, 0.00050, 2.22904)	(-1.01067, 2.11833, 0.22584)	(0.93965,1.98359, 0.26206)
$(s_1^i,\ s_2^i,\ s_3^i)$	(5.98263, 0.04161, 0.00031)	(3.55393, 0.25155, 1.49865)	(0.00085 4.44834, 6.99102)

TABLE III
PERFORMANCE OF SEIT2FNN AND OTHER TYPE-2 FUZZY SYSTEMS IN EXAMPLE 1

Methods	T2FLS (Singleton)	T2FLS (TSK)	SEIT2FNN*	SEIT2FNN**
Rule number	15	10	10	10
Parameter number	120	120	120	90
Iterations	500	500	500	500
RMSE	0.0306	0.0217	0.0014	0.0017

*Without fuzzy set reduction.
** With fuzzy set reduction.

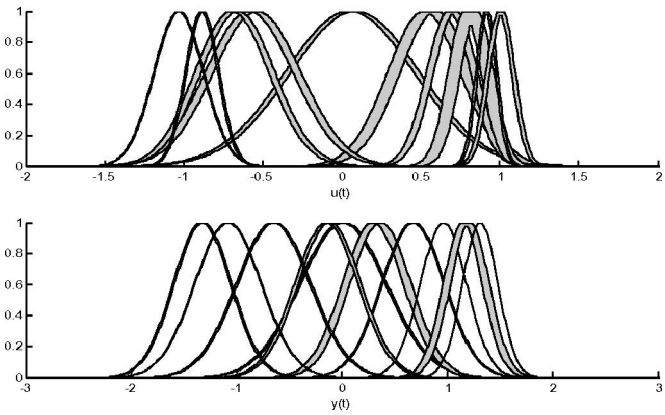

Fig. 3. Distributions of the type-2 fuzzy sets in inputs $u(k)$ and $y(k)$ without fuzzy set reduction operations in Example 1.

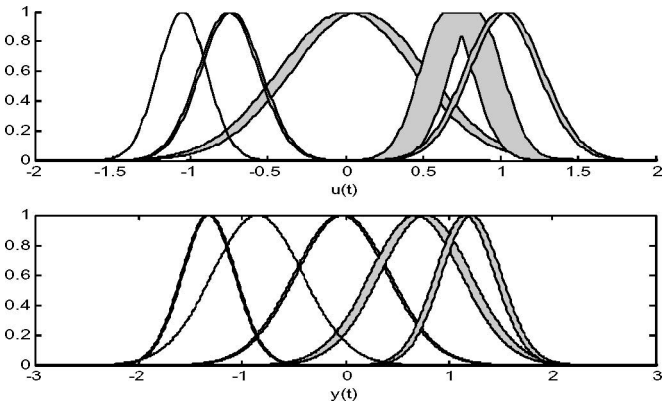

Fig. 4. Distributions of the type-2 fuzzy sets in the $u(k)$ and $y(k)$ dimensions with fuzzy set reduction operations in Example 1.

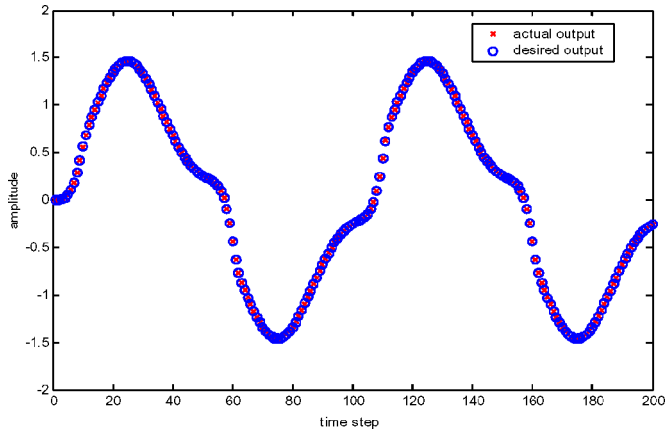

Fig. 5. Identification results of the SEIT2FNN using ten rules in Example 1.

B. Example 2—Time Varying Plant Identification

To illustrate the SEIT2FNN's ability to evolve, the plant in Example 1 is modified to be

$$y_d(t+1) = \frac{y_d(t)}{1+y_d^2(t)} + u^3(t) + f(t) \tag{43}$$

where

$$f(t) = \begin{cases} 0, & 1 \le t \le 1000 \\ 1.0, & 1001 \le t \le 2000 \\ 0, & 2001 \le t. \end{cases} \tag{44}$$

That is, the plant parameter changes with time. As in Example 1, the input is $u(t) = \sin(2\pi t/100)$. In this example, the SEIT2FNN is used to identify the plant online, i.e., no data are collected in advance for offline training. The SEIT2FNN learning coefficients are the same as those used for Table I in Example 1. Fig. 6 shows the SEIT2FNN learning errors, where each plotted value denotes the accumulated square error over 100 adjacent time steps. Fig. 7 shows the corresponding learning results of the SEIT2FNN. Initially, there are no rules in the SEIT2FNN, and three rules are generated after 1000 time steps of online learning. The system model changes after 1000 time steps, and one additional rule is generated to cover these new inputs. Due to the online structure learning ability, Fig. 6 shows that there is no abrupt change in the error. After 2000 time steps, the plant model changes again. Since the plant model is the same as that in time steps 1–1000, no new rules are generated. The SEIT2FNN adapts to this plant change by parameter learning only.

TABLE IV
INFLUENCE OF η ON THE PERFORMANCE OF SEIT2FNN WITH SIX RULES WHEN $\phi_{\rm th} = 0.05$ IN EXAMPLE 1

η	0.001	0.01	0.05	0.1	0.15	0.2	0.3
Test RMSE	0.00855	0.00510	0.00218	0.00173	0.001597	0.00170	0.00163

TABLE V
INFLUENCE OF $\phi_{\rm th}$ ON THE PERFORMANCE OF SEIT2FNN WHEN $\eta = 0.05$ IN EXAMPLE 1

ϕ_{th}	0.01	0.025	0.05	0.075	0.15	0.2
Rules	3	5	6	7	10	14
Test RMSE	0.0062	0.00278	0.00218	0.00206	0.0014	0.0017

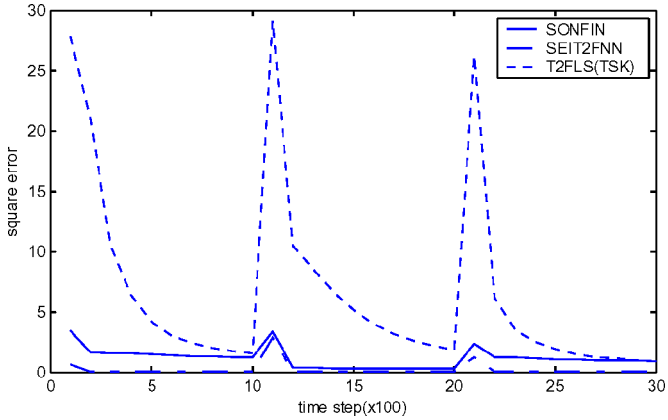

Fig. 6. Online learning errors of the first time-varying plant identification problem in Example 2 using different fuzzy neural networks.

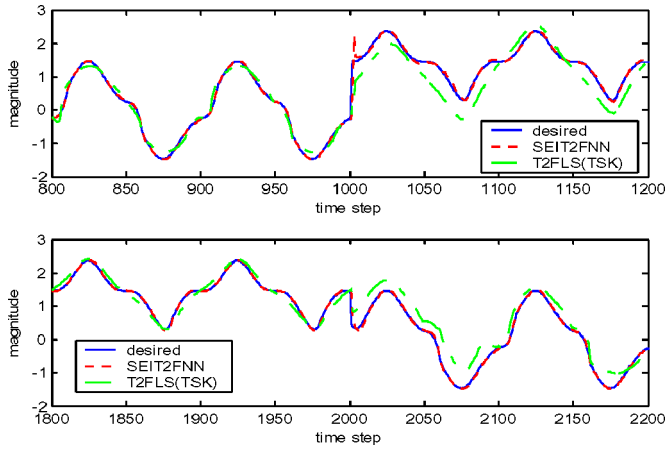

Fig. 7. Online identification results of the first time-varying plant in Example 2 using the SEIT2FNN.

For comparison, the TSK-type T2FLS is applied to the same problem. The number of rules in the T2FLS is set *a priori* at 3. Fig. 6 shows the T2FLS learning errors, and indicates that an abrupt increase in error occurs after time step 1000, as there is no structure evolution ability in T2FLS. A similar phenomenon occurs after time step 2000. The reason for this is that existing cluster locations have moved to cover the training data pairs in time steps 1001–2001. However, when the data pairs become similar to the original data pairs in time steps 1–1001, these new locations initially achieve poor performance. Fig. 7 shows the T2FLS learning results, and indicates that the SEIT2FNN performance is better than the T2FLS performance at each time step.

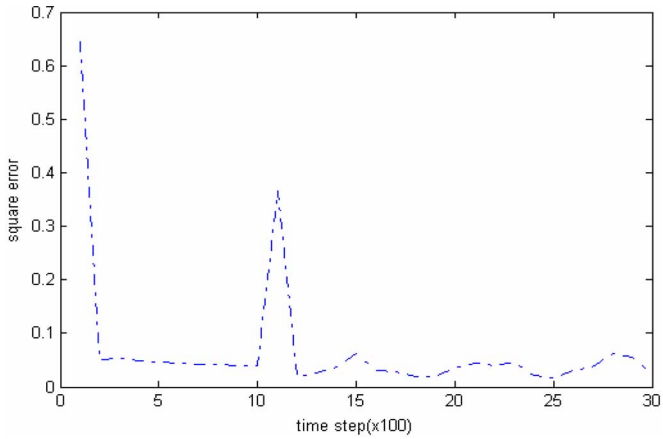

Fig. 8. Online learning errors of the second time-varying plant in Example 2 using the SEIT2FNN.

The performance of the SONFIN, which also has online structure learning abilities, is also compared. The generated number of rules in the SONFIN in time steps 1–1000 is 7, and increases to 16 after time step 1000 due to structure learning ability. Fig. 7 shows the SONFIN learning errors. Since the SONFIN also has structure learning abilities, the error change is small after time step 1000. However, this result is achieved by an increment of nine type-1 rules in contrast to only one type-2 rule in the SEIT2FNN.

Identification of another time-varying plant with a time-varying parameter by SEIT2FNN was also performed in this example. The plant is also a modification of the plant in Example 1 and is described by

$$y_d(t+1) = \frac{\alpha(t) y_d(t)}{1 + y_d^2(t)} + u^3(t) \tag{45}$$

where

$$\alpha(t) = \begin{cases} 1, & 1 \le t \le 1000 \\ \sin(t/100), & 1000 < t. \end{cases} \tag{46}$$

As in Example 1, the input is $u(t) = \sin(2\pi t/100)$. As in the earlier time-varying plant, three rules are generated after 1000 time steps of online learning. The system parameter changes continuously after 1000 time steps, and five additional rules are generated to cover these new inputs. Fig. 8 shows the SEIT2FNN learning errors for $t = 1, \ldots, 3000$, where each plotted value denotes the accumulated square error over 100 adjacent time steps. Fig. 9 shows the learning results of the SEIT2FNN. The results show that SEIT2FNN can online model this time-varying plant with good performance.

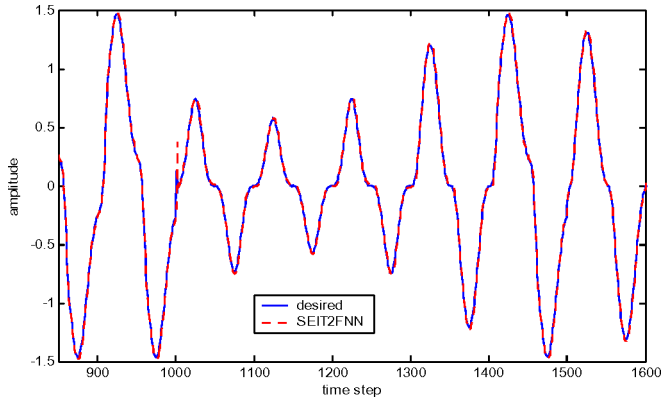

Fig. 9. Online identification results of the second time-varying plant in Example 2 using the SEIT2FNN.

C. Example 3—System Identification With Noise

This example is conducted to illustrate the SEIT2FNN's noise resistance ability. This example identifies the same plant in Example 1, except that it is assumed that the measured value y_d contains noise. The added noise is artificially generated white noise with uniform distribution in $[-2.5, 2.5]$. In a realistic situation, the noise may be colored noise. This example only simulates uniform white noise, as was studied in previous literatures [3], [38]. Identification is performed online from time step $t = 1$ to $t = 1000$ without repeated training of each sample. The SEIT2FNN learning coefficients are the same as those used for Table I in Example 1. There are 20 Monte Carlo realizations. On average, three rules are generated for the SEIT2FNN. Table VI shows the learned antecedent and consequent parameters in the three rules of SEIT2FNN. Fig. 10 shows the average online learning performance of the SEIT2FNN over these 20 realizations, where each plotted value denotes accumulated square errors over 100 adjacent time steps. The error is defined as the difference between the SEIT2FNN output and actual desired output without noise. Fig. 11 shows one learning result of the 20 realizations, showing that the SEIT2FNN output does not change as violently as the actual noisy output. This result verifies the SEIT2FNN's noise resistance abilities.

The self-evolving type-1 system, SONFIN, is also applied to the same problem. Fig. 10 shows the average learning errors of the SONFIN, with an average of seven rules in the SONFIN. These results show that the SEIT2FNN learning performance is better than the SONFIN learning performance when there is noise.

D. Example 4—Adaptive Noise Cancellation

This example applies the SEIT2FNN to the adaptive noise cancellation problem that was also studied in [38] and [39]. Adaptive noise cancellation is concerned with the enhancement of noise corrupted signals. It is based on the availability of a primary input source and an auxiliary (reference) input source that is located at the noise field and contains no or little signal (as shown in Fig. 12). In Fig. 12, the primary input source contains the desired signal s, which is corrupted by noise n_0 generated from the noise source n. The received signal is thus

$$x(t) = s(t) + n_0(t). \tag{47}$$

The secondary or auxiliary (reference) input source receives the noise n_1, which is correlated with the corrupting noise n_0. The principle of the adaptive noise cancellation techniques is to adaptively process the reference noise n_1 to generate a replica of n_0, and then subtract the replica of n_0 from the primary input x to recover the desired signal s. We denote the replica of n_0 as process y, which is the output of the SEIT2FNN. The assumptions that s, n_0, and n_1 are stationary zero-mean processes, s is uncorrelated with n_0 and n_1, and n_0 and n_1 are correlated, are made. Also, the reference input source is situated in such a position that it detects only the noise, not the signal s. From Fig. 12, we have

$$e(t) = s(t) + n_0(t) - y(t). \tag{48}$$

As in [38], the primary input source is $s(t) = 0.6\sin(0.06t)\cos(0.01t)$, and the noise source n is white noise with uniform distribution in $[-1.5, 1.5]$. Assume that the relation between the noise source n and the corrupting noise n_0 is a nonlinear function

$$n_0(t) = 0.6(n(t))^3 \tag{49}$$

and the reference input is placed in front of the noise source so that $n_1(t) = n(t)$. The input of the SEIT2FNN is $n_1(t)$ and the output is $y(t)$. The learning coefficient η is set at 0.01. The structure-learning threshold ϕ_{th} is set at about 0.1. Since there is only one input in the SEIT2FNN, a rule corresponds to a fuzzy set, and fuzzy set reduction operation is unnecessary. The SEIT2FNN conducts online learning. Fig. 13 shows the online learning result from time step $k = 48\,000$–$50\,000$, where Fig. 13(a)–(c) shows the original, corrupted, and recovered signal. Initially, the SEIT2FNN has no rules in it, and generates seven rules during the online training process.

The online learning performance of the SEIT2FNN is compared to the T2FLS and adaptive neural fuzzy network (ANFF) [38]. The ANFF is a type-1 fuzzy system with online structure-learning ability. During the online learning process, ANFF generates seven rules. A TSK-type T2FLS is simulated, and the number of rules is also set to be seven *a priori*. The square errors between the recovered and original signals are computed for each network. Fig. 14 shows the online learning error of different methods, where the error is calculated from the start of learning, and each plotted error value is the sum of square errors over 60 adjacent time steps. The results show that with the same number of fuzzy rules, the online learning performance of the two type-2 fuzzy systems (SEIT2FNN and T2FLS) is better than the type-1 fuzzy system ANFF. For the two type-2 fuzzy systems, the performance of the SEIT2FNN is better than the T2FLS.

E. Example 5—Prediction of a Chaotic Time Series

The time series prediction problem used in this example is the chaotic Mackey–Glass chaotic time series that is generated

TABLE VI
LEARNED PARAMETERS IN SEIT2FNN WITH THREE RULES IN EXAMPLE 3

	Rule 1	Rule 2	Rule 3
$(m_{11}^i, m_{21}^i, \sigma_1^i)$	(-0.04678, -0.04270, 0.63399)	(0.57327, 0.92256, 0.366061	(-0.94523, -0.86596, 1.09667)
$(m_{12}^i, m_{22}^i, \sigma_2^i)$	(-0.12937, -0.12106, 0.55606)	(0.55555, 0.63948, 0.75062)	(-0.87301, -0.56835, 1.05721)
$(c_1^i,\ c_2^i,\ c_3^i)$	(-0.33736, 0.00050, 1.58049)	(-0.81564, 1.85217, 0.26247)	(0.97397, 2.28592, 0.09994)
$(s_1^i,\ s_2^i,\ s_3^i)$	(0.64622, 3.06998, 0.12041)	(0.97260, 0.11999, 1.96588)	(0.73782, 1.32333, 1.64623)

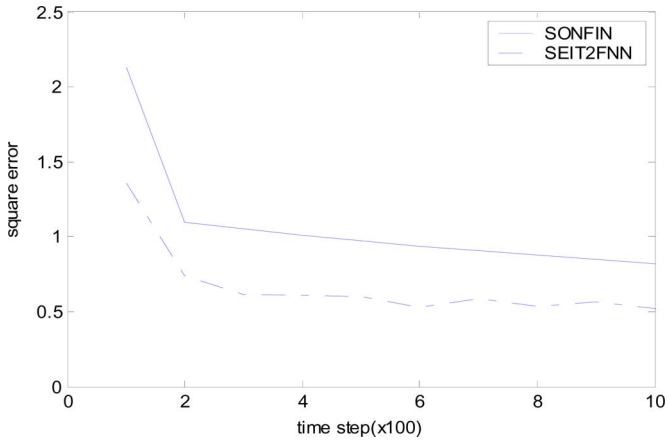

Fig. 10. Average online learning errors of the noisy plant identification problem in Example 3 using different fuzzy neural networks.

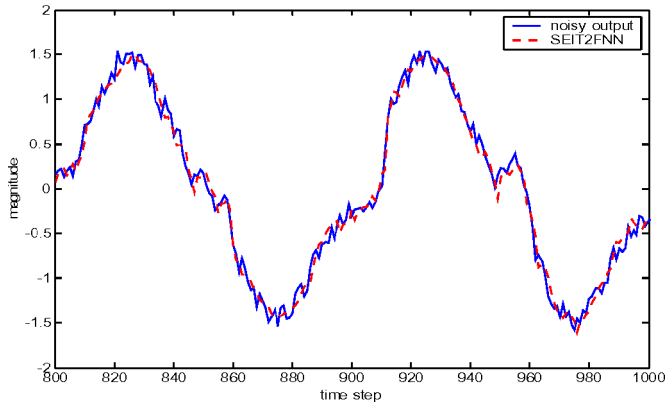

Fig. 11. Online learning results of the SEIT2FNN when noise appears in the measured data in Example 3.

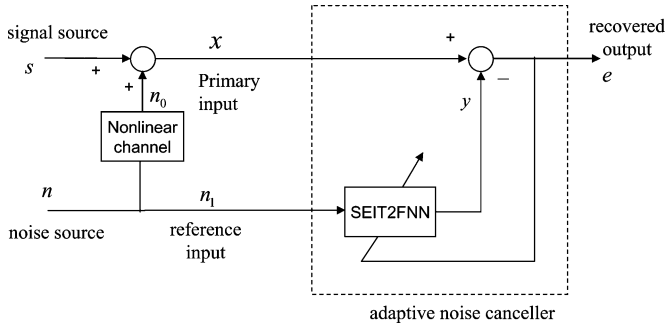

Fig. 12. Block diagram of adaptive noise cancellation using the SEIT2FNN.

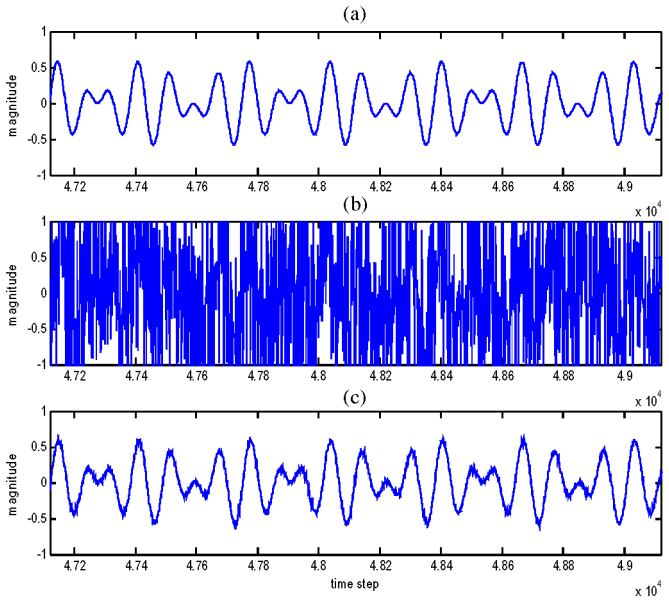

Fig. 13. (a) Original signal source s. (b) Corrupted signal x. (c) Recovered signal using the SEIT2FNN in Example 4.

from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{50}$$

where $\tau > 17$. As in previous studies [40]–[47], parameter τ is set to be 30 and $x(0) = 1.2$. Four past values are used to predict $x(t)$, and the input–output data format is

$$[x(t-24), x(t-18), x(t-12), x(t-6); x(t)].$$

One thousand patterns were generated from $t = 124$ to $t = 1123$, with the first 500 patterns being used for training and the last 500 for testing. The parameters in the SEIT2FNN are set to be $\phi_{\text{th}} = 0.11$, $\rho = 1$, and $\eta = 0.05$. After 1000 epochs of training, seven rules are generated. Table VII shows the test performance and Fig. 15 shows the prediction results. To further reduce the number of fuzzy sets, parameter $\phi_{\text{th}} = 0.29$ and $\rho = 0.9$ are chosen. After 1000 epochs of training, seven rules are also generated. However, there are only 5, 6, 2, and 3 fuzzy sets in inputs $x(t-24)$, $x(t-18)$, $x(t-12)$, and $x(t-6)$, respectively. Table VII shows the test performance. The result shows that though the number of fuzzy sets is reduced from 28 to 16, the increment in the test error is small.

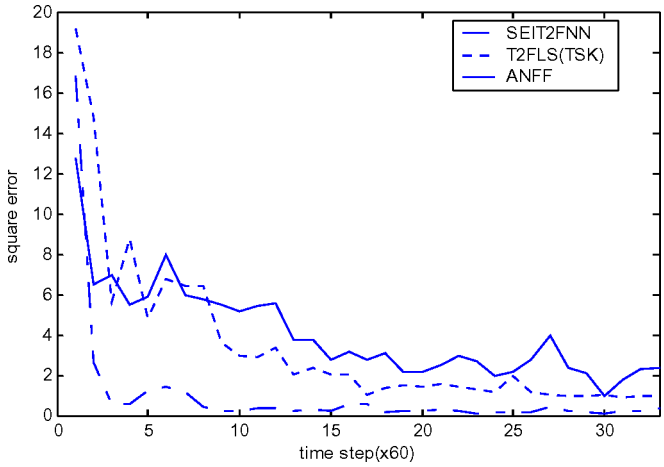

Fig. 14. Comparison of square errors between the recovered signal and original signal by different methods in Example 4.

TABLE VII
PERFORMANCE OF SEIT2FNN AND OTHER TYPE-2 FUZZY SYSTEMS IN EXAMPLE 5

Methods	T2FLS (Singleton)	T2FLS (TSK)	SEIT2FNN*	SEIT2FNN**
Rule number	16	7	7	7
RMSE	0.0426	0.0431	0.0034	0.0053

*Without fuzzy set reduction.
** With fuzzy set reduction.

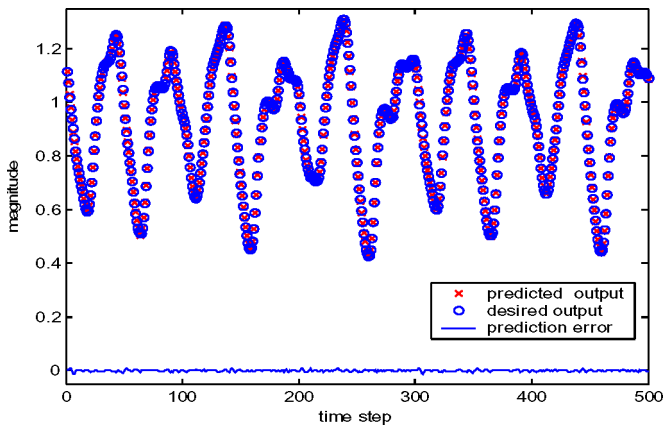

Fig. 15. Test result of the chaotic series prediction problem using the SEIT2FNN with seven rules in Example 5.

Prediction by the T2FLS with singleton and TSK-type consequents are also conducted. Table VII shows the number of rules and prediction errors. The result shows that for similar prediction performance, the number of rules in the TSK-type T2FLS is smaller than the singleton-type T2FLS. With the same number of rules, the prediction error of the SEIT2FNN is much smaller than the TSK-type T2FLS, verifying the powerful learning ability of SEIT2FNN in constructing type-2 systems.

The performance of SEIT2FNN is compared with some recently developed type-1 fuzzy systems designed by genetic algorithms or neural learning, including the radial basis function-based adaptive fuzzy system (RBF-AFS) [40], hybrid neural fuzzy inference system (HyFIS) [41], neurofuzzy function approximator (NEFPROX) [42], dynamic FNN (D-FNN) [43], a combination of the genetic algorithm and gradient descent algorithm (GEFREX) [44], subsethood-product fuzzy neural inference system (SuPFuNIS) [45], generalized FNN (G-FNN) [46], and clustering-aided simplex particle swarm optimization (CSPSO) with four and ten (27 fuzzy sets in total) rules [47]. Table VIII shows the performance of these compared type-1 fuzzy systems, indicating that the SEIT2FNN achieves smaller prediction error, and the number of rules in the SEIT2FNN is also smaller than these compared type-1 fuzzy systems.

TABLE VIII
PERFORMANCE OF SEIT2FNN AND OTHER TYPE-1 FUZZY SYSTEMS IN EXAMPLE 5

Methods	Rule number	RMSE
RBF-AFS [40]	21	0.0128
HyFIS [41]	16	0.01
NEFPROX [42]	26	0.0533
D-FNN [43]	10	0.008
GEFREX [44]	20	0.0067
SuPFuNIS [45]	10	0.0057
G-FNN [46]	10	0.0056
CSPSO [47]	4	0.0068
CSPSO [47]	10	0.0064
SEIT2FNN	7	0.0034

V. CONCLUSION

This paper proposed a new type-2 fuzzy system, the SEIT2FNN. In contrast to existing type-2 fuzzy systems, there is no need to determine SEIT2FNN structure in advance because the proposed structure-learning ability enables the SEIT2FNN to evolve its structure online. This self-evolving property is especially important for time-varying systems or systems whose operating point changes with time. For the time-varying systems identification problem, simulation results showed that using only online parameter learning cannot efficiently handle the variation. The online structure evolution ability in SEIT2FNN helps evolve new rules to handle this time-varying phenomenon in an efficient way, and good performance is achieved. Moreover, the proposed rule-ordered Kalman filter algorithm helps tune the consequent parameters online and improves learning accuracy. Simulation results showed that, compared with type-1 fuzzy systems that also have online structure-learning ability, the SEIT2FNN achieves higher learning accuracy with a smaller number of rules for both clean and noisy data. The major reason is that the interval type-2 fuzzy sets in the antecedent and consequent part of SEIT2FNN add extra degrees of freedom for SEIT2FNN learning. Since these additional flexible parameters provided by the footprint of uncertainty (FoU) are interpretable, we can assign their appropriate initial values according to their meaning in terms of fuzzy sets during structure learning. Such an appropriate initial assignment helps improve the learning speed and overcome the learning burden problem that usually comes up with additional parameter flexibility. This explains the reason why SEIT2FNN shows better performance than type-1

FNNs even for clean data, and interval type-2 fuzzy systems with only parameter learning may have poorer performance than type-1 FNNs with structure learning. In addition, the FoU in SEIT2FNNs helps handle the numerical uncertainty associated with system inputs and outputs. Therefore, SEIT2FNN has the potential to achieve better performance than type-1 fuzzy systems when dealing with noisy data, as was shown in the simulation examples. Future studies will theoretically analyze the learning convergence of SEIT2FNNs and examine practical applications of the SEIT2FNN to control and signal processing problems with colored noise or uncertainty.

REFERENCES

[1] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, Dec. 1999.

[2] J. M. Mendel and R. I. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 117–127, Apr. 2002.

[3] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic System: Introduction and New Directions*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[4] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 551–563, Oct. 2000.

[5] H. B. Mitchell, "Pattern recognition using type-2 fuzzy sets," *Inf. Sci.*, vol. 170, pp. 409–418, 2005.

[6] P. Melin and O. Castillo, "Intelligent control of non-linear dynamic plants using type-2 fuzzy logic and neural networks," presented at the IEEE Int. Conf. Fuzzy Syst., Budapest, Hungary, Jul. 2004.

[7] H. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, Aug. 2004.

[8] M. Melgarejo and C. Pena-Reyes, "Hardware architecture and FPGA implementation of a type-2 fuzzy system," in *Proc. Great Lakes Symp. VLSI (GLSVLSI)*, Boston, MA, 2004, pp. 458–261.

[9] R. I. John, P. R. Innocent, and M. R. Barnes, "Neuro-fuzzy clustering of radiographic tibia image data using type-2 fuzzy sets," *Inf. Sci.*, vol. 125, pp. 203–220, 2000.

[10] A. G. Luigi Di Lascio and A. Nappi, "Medical differential diagnosis through type-2 fuzzy sets," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2005, pp. 371–376.

[11] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems,* ch. 19. Englewood Cliffs, NJ: Prentice-Hall, May 1996.

[12] G. Mouzouris and J. M. Mendel, "A singular-value-QR decomposition based method for training fuzzy systems in uncertain environments," *J. Intell. Fuzzy Syst.*, vol. 5, pp. 367–374, 1997.

[13] J. Yen and L. Wang, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Trans. Syst., Man, Cyber. B*, vol. 29, no. 1, pp. 13–24, Feb. 1999.

[14] J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May 1993.

[15] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.

[16] C. S. Ouyang, W. J. Lee, and S. J. Lee, "A TSK-type neurofuzzy network approach to system modeling problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 751–767, Aug. 2005.

[17] D. Kukolj and E. Levi, "Identification of complex systems based on neural and Takagi-Sugeno fuzzy model," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 272–282, Feb. 2004.

[18] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[19] P. P. Angelov and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

[20] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, "Similarity measures in fuzzy rule base simplification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 3, pp. 376–386, Jun. 1998.

[21] Y. Jin, "Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 2, pp. 212–221, Apr. 2000.

[22] H. Wang, S. Kwong, Y. Jin, W. Wei, and K. Man, "A multi-objective hierarchical genetic algorithm for interpretable rule-based knowledge extraction," *Fuzzy Sets Syst.*, vol. 149, no. 1, pp. 149–186, 2005.

[23] R. Paiva and A. Dourado, "Interpretability and learning in neuro-fuzzy systems," *Fuzzy Sets Syst.*, vol. 147, no. 1, pp. 17–38, 2004.

[24] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 535–550, Oct. 2000.

[25] C. H. Lee, Y. C. Lin, and W. Y. Lai, "Systems identification using type-2 fuzzy neural network (Type-2 FNN) systems," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 2003, vol. 3, pp. 1264–1269.

[26] C. H. Wang, C. S. Cheng, and T. T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1462–1477, Jun. 2004.

[27] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic system," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, Feb. 2004.

[28] H. Hagras, "Comments on dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 36, no. 5, pp. 1206–1209, Oct. 2006.

[29] G. M. Mendez and O. Castillo, "Interval type-2 TSK fuzzy logic systems using hybrid learning algorithm," in *Proc. IEEE Int. Conf. Fuzzy Syst.,* May 22–25, 2005, pp. 230–235.

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

[31] V. V. Cross and T. A. Sudkamp, *Similarity and Compatibility in Fuzzy Set Theory: Assessment and Application*. Heidelberg, Germany: Physica-Verlag, 2002.

[32] D. Wu and J. M. Mendel, "A vector similarity measure for interval type-2 fuzzy sets," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2007, pp. 1–6.

[33] L. Guo, "Estimating time-varying parameters by the Kalman filter based algorithm: Stability and convergence," *IEEE Trans. Autom. Control*, vol. 35, no. 2, pp. 141–147, Feb. 1990.

[34] J. J. Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm," *Neurocomputing*, vol. 70, pp. 2460–2466, 2007.

[35] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (Advances in Fuzzy Systems – Applications and Theory),* vol. 19. Singapore: World Scientific, 2001.

[36] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimentional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[37] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, Nov. 1997.

[38] C. T. Lin and C. F. Juang, "An adaptive neural fuzzy filter and its applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 4, pp. 635–656, Aug. 1997.

[39] O. Castillo and P. Melin, "Adaptive noise cancellation using type-2 fuzzy logic and neural networks," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2004, vol. 2, pp. 1093–1098.

[40] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification," *Fuzzy Sets Syst.*, vol. 83, pp. 325–339, 1996.

[41] J. Kim and N. K. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamic systems," *Neural Netw.*, vol. 12, pp. 1301–1319, 1999.

[42] D. Nauk and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets Syst.*, vol. 101, no. 2, pp. 261–271, 1999.

[43] S. Wu and M. J. Er, "Dynamic fuzzy neural networks—A novel approach to function approximation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 358–364, Apr. 2000.

[44] M. Russo, "Genetic fuzzy learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 259–273, Sep. 2000.

[45] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 578–599, May 2002.

[46] Y. Gao and M. J. Er, "NARMAX time series model prediction: Feedforward and recurrent fuzzy neural approaches," *Fuzzy Sets Syst.*, vol. 150, pp. 331–350, 2005.

[47] C. F. Juang, I. F. Chung, and C. H. Hsu, "Automatic construction of feedforward/recurrent fuzzy systems by clustering-aided simplex particle swarm optimization," *Fuzzy Sets Syst.*, vol. 158, no. 18, pp. 1979–1996, Sep. 2007.

Chia-Feng Juang (M'00–SM'08) received the B.S. and Ph.D. degrees in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1993 and 1997, respectively.

In 2001, he joined the National Chung Hsing University (NCHU), Taichung, Taiwan, where he is currently a Professor of electrical engineering. He has authored or coauthored 100 refereed journal and conference papers and three book chapters in the area of computational intelligence, and has been a referee for about 40 international journals. Over 460 journal papers have cited his published papers during the last ten years (interservices intelligence (ISI) Web of Science). His current research interests include computational intelligence, intelligent control, computer vision, speech signal processing, and field-programmable gate arrays (FPGA) chip design.

Dr. Juang currently serves on the Editorial Board of the *Open Cybernetics and Systemics Journal* and the *International Journal of Computational Intelligence in Control*, and is an Area Editor of the *International Journal of Intelligent Systems Science and Technology*. He is a Senior Member of the IEEE Computational Intelligence Society and the Systems, Man, and Cybernetics Society. He was the recipient of the Youth Automatic Control Engineering Award from the Chinese Automatic Control Society, Taiwan, in 2006, and the Outstanding Youth Teacher Award from NCHU, Taiwan, in 2007.

Yu-Wei Tsao received the B.S. degree in electrical engineering from the National Chung-Hsing University, Taichung, Taiwan, R.O.C., in 2007.

In 2007, he joined the Industrial Technology Research Institute, Hsinchu, Taiwan. His current research interests include type-2 neural fuzzy systems networks and field-programmable gate arrays (FPGA) chip design.