Simplified Interval Type-2 Fuzzy Neural Networks

Yang-Yin Lin, Shih-Hui Liao, Jyh-Yeong Chang, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*— This paper describes a self-evolving interval type-2 fuzzy neural network (FNN) for various applications. As type-1 fuzzy systems cannot effectively handle uncertainties in information within the knowledge base, we propose a simple interval type-2 FNN, which uses interval type-2 fuzzy sets in the premise and the Takagi–Sugeno–Kang (TSK) type in the consequent of the fuzzy rule. The TSK-type consequent of fuzzy rule is a linear combination of exogenous input variables. Given an initially empty the rule-base, all rules are generated with on-line type-2 fuzzy clustering. Instead of the time-consuming K-M iterative procedure, the design factors q_l and q_r are learned to adaptively adjust the upper and lower positions on the left and right limit outputs, using the parameter update rule based on a gradient descent algorithm. Simulation results demonstrate that our approach yields fewer test errors and less computational complexity than other type-2 FNNs.**

***Index Terms*— Fuzzy identification, on-line fuzzy clustering, type-2 fuzzy neural networks (FNNs), type-2 fuzzy systems.**

I. Introduction

THE MARRIAGE of fuzzy logic systems (FLSs) and neural networks (NNs) has drawn considerable attention in recent years. So-called fuzzy neural networks (FNNs) [1]–[8], [40]–[42], [51]–[53] inherit their learning ability from NNs, and much of their inference technology from fuzzy systems, which are widely used in robotics, temperature control, system identification, bioengineering, and many others. Some FNNs, including, for example, the adaptive network-based fuzzy inference system (ANFIS) [1], the fuzzy adaptive learning control network [3] and the self-constructing neural fuzzy inference network (SONFIN) [4], are well known. ANFIS uses a fixed structure, with all parameters turned by a hybrid learning algorithm. For the consequent part of their fuzzy rule, ANFIS and SONFIN use the Takagi–Sugeno–Kang (TSK) type and Mamdani-type, respectively. Many studies [1], [2], [4], [5] indicate that TSK-type FNNs achieve superior learning accuracy than Mamdani-type FNNs.

Manuscript received July 17, 2013; revised September 10, 2013; accepted October 1, 2013. Date of publication October 17, 2013; date of current version April 10, 2014. This work was supported in part by the Aiming for the Top University Plan of National Chiao Tung University, in part by the Ministry of Education of Taiwan under Grant 102W963, and in part by the UST-UCSD International Center of Excellence in Advanced Bio-Engineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant NSC-101-2911-I-009-101.

Y.-Y. Lin is with the Institute of Electrical Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: oliver.yylin@gmail.com).

S.-H. Liao and J.-Y. Chang are with the Institute of Electrical Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: liao622@gmail.com; jychang@mail.nctu.edu.tw).

C.-T. Lin is with the Institute of Electrical Control Engineering and Brain Research Center, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: ctlin@mail.nctu.edu.tw).

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/TNNLS.2013.2284603

In the real world, most plant operations are exposed to a variety of internal and external disturbances, and so must apply time-varying techniques to contend with uncertainty. Unfortunately, the aforementioned FNNs are unable to directly handle rule uncertainties, since type-1 fuzzy sets are precise. To address this deficiency, Zadeh [9] introduced type-2 fuzzy sets as an extension of the type-1 fuzzy set. Judging from the relevant literature [10]–[21], [34], [36], [47], the use of type-2 fuzzy sets in diverse applications shows that type-2 methods have more potential than type-1 methods in coping with uncertainties, such as noisy measurements, variations in word meanings, and so on. Specifically, type-2 FLSs allow researchers to model and minimize the effect of uncertainties associated with rule-base systems; this capability, however, does incur the additional computational cost of performing type reduction from type-2 to type-1. In an effort to reduce this additional cost, in [10]–[15], a complete type-2 FLS theory has been established, including a full formalization of type-reduction. We consider this paper critical since, in the past few years, most of the type-2 FNNs researchers [21]–[31], [33], [36], [43]–[46], [48]–[50], [54] have focused exclusively on interval type-2 fuzzy systems.

In [23], Lee *et al*. presented the design of type-2 FNN for nonlinear system processing. In [25], Wang *et al*. provided parameter learning algorithms for interval type-2 fuzzy systems using Gaussian primary membership functions (MFs) with uncertain means as type-2 fuzzy sets. In a further study [27], the self-evolving interval type-2 FNN (SEIT2FNN) is applied to system identification and channel equalization, using type-2 Gaussian MFs with uncertain means and fixed standard deviation (STD) in the antecedent part, and demonstrating simultaneous structure and parameter learning capability. Abiyev and Kaynak [30] proposed a novel type-2 TSK-based fuzzy neural structure (FNS) that used the q coefficient to adjust the proportion of the upper and lower bound in terms of the final output. A recent study [29] shows that the type-2 FNN uses two forms of interval type-2 fuzzy sets, one with uncertain means but a fixed STDs and the other with uncertain STDs but fixed mean, and compared their performance. The structure of a discrete interval type-2 fuzzy system by fuzzy c-means clustering was presented in [32]. In [33], Lee *et al*. reported that the use of interval asymmetric type-2 fuzzy sets to construct FNNs is effective for identification of nonlinear dynamic systems.

In all of the aforementioned type-2 FNNs except the TSK T2 FNS [30], the consequent weights need to be rearranged in ascending order according to the K-M iterative procedure for finding L and R end points. Thus, their execution time will increase with the number of rules. This paper presents a simplified interval type-2 neural fuzzy network (SIT2FNN)

that can be formulated without incurring such computational costs. The type-reduction process in the SIT2FNN capitalizes control factors ql and qr instead of K-M iterative method to overcome time-consuming because of arranging consequent part parameters. Using a gradient descent algorithm to adapt systems, the control factors enable to adjust the proportion of the upper and lower portions in terms of the left and right bounds. The premise clause of fuzzy rule in SIT2FNN is characterized by an interval type-2 fuzzy set, and the consequent part is the TSK-type with an interval set. The SIT2FNN possesses self-evolving in that it can automatically grow the required network structure (rule number and initial fuzzy set shape) and parameters according to their training data. The SIT2FNN can also learn parameters using an algorithm in which all free weights are derived using a gradient descent algorithm. Several simulations are conducted to verify SIT2FNN performance. These simulations are also compared with other type-1 and type-2 systems, demonstrating that the SIT2FNN could achieve superior performance for noise-free or noisy environment.

The rest of this paper is organized as follows. Section II gives a brief survey of some existing methods, Section III introduces the overall SIT2FNN structure, and Section VI describes the structure learning of the SIT2FNN and its parameter update rules. Section V presents four simulation examples, including system identification, and prediction of chaotic and Santa Fe time-series. Section VI provides the concluding remarks.

II. Brief Survey of Some Existing Methods

In recent years, considerable research has been devoted toward these developing interval type-2 FNNs (IT2FNNs) [25]–[29], [31], [33], [34], [36], [43]–[50], [54] for addressing real world problems regarding uncertainty and imprecise. In [25], Wang *et al.* present an IT2FNN for handling uncertainty with dynamical optimal learning. However, a fixed structure in the IT2FNN could not effectively address upon time-varying systems. Afterward, many studies [4], [27]–[29], [31], [34]–[36], [40]–[41], [49], [54] are presented using self-evolving property that can automatically evolve the required structure and parameters according to piece of training data to time-varying systems. However, one of the well-known interval type-2 FNN (SEIT2FNN) [27] is proposed for modeling and identification of time-varying systems, and is useful for noise tolerance in terms of the corrupt data. The consequent part in the SEIT2FNN is of TSK-type fuzzy reasoning, which is linear model of input variables with interval. Unlike the consequent part of the SEIT2FNN, the functional-link interval type-2 compensatory FNN [28] employs the functional-link NN to construct then part to providing greater discrimination capability in the input pattern space and uses compensatory operation to adaptively adjust fuzzy inference between the t-norm and t-conorm. Juang *et al.* [29] presented recurrent structure in the IT2FNN for dynamic system processing. The recurrent structure not only depends on the previous states but on the current states, and thus, it seems to be more precise than only considering the prior states. In [36], a self-organizing neural fuzzy system with Mamdani-type fuzzy reasoning that is insensitive to noisy environments is presented.

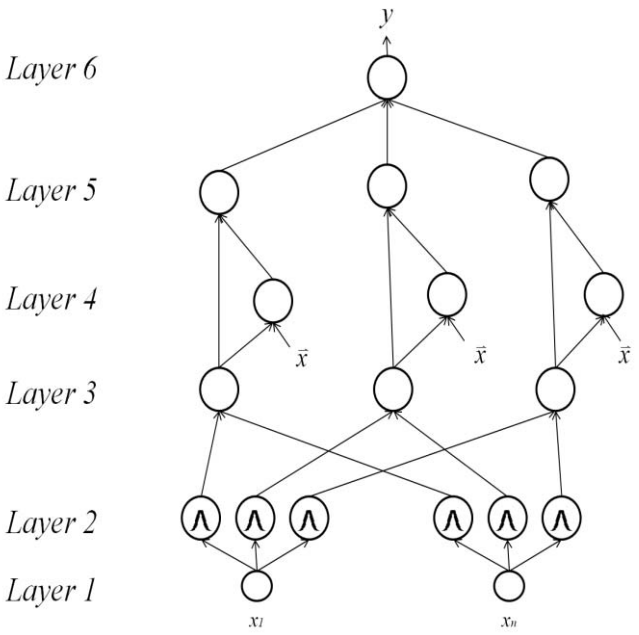

Fig. 1. Structure of SIT2FNN, where each node in layer 4 performs a linear combination of input states with intervals.

To update the consequent part parameters, several studies [27], [29], [44] used the Kalman filter algorithm to strength network performance and fostered the learning effort. In [43], a novel homogeneous integration strategy of an interval type-2 FIS (IT2FIS) with hybrid learning is presented. All of the aforementioned IT2FNNs might lead to computational cost in terms of the type reduction process. Therefore, Abiyev and Kaynak [30] reported that the use of control factor q, which can adjust the proportion of the upper and lower bounds to reduce computational burden. TSK T2 FNS uses the q factor to decrease the processing time, but its method seems to reduce the degrees of freedom with more guided consequent parameters. Biglarbegian *et al.* [50] demonstrated the stability of interval type-2 TSK fuzzy logic control systems. For the hybrid learning manner of IT2FIS [44], premise and consequent parameters are tuned by a recursive least square and a gradient descent algorithm, respectively. Recently, the use of bioinspired optimization methods in the SIT2FNN has presented for tackling the complex task of finding the optimal parameters and structure of fuzzy systems. Castillo and Melin [45] concisely introduced the optimization of type-2 fuzzy systems based on bioinspired methods.

The bioinspired optimization algorithms, like GA [25], [46], [47], PSO [48], and ACO [49], are less likely to be stuck in a local minimum, but their execution time is time-consuming. Thus, in this paper, the combination of self-evolving property and control factors to simplify conventional IT2FNNs is presented.

III. SIT2FNN Structure

This section introduces the structure of a multiple-input-single-output SIT2FNN. Fig. 1 shows the proposed six-layered

SIT2FNN structure. The premise parts of the SIT2FNN engage interval type-2 fuzzy sets characterized with uncertain means and fixed STDs, and the consequent part of fuzzy rule is of a TSK type that executes a linear combination of input variables with an interval set. Each SIT2FNN rule can be denoted as

$$\begin{aligned}&\text{Rule } i\text{: IF } x_1 \text{ is } \tilde{A}_1^i \text{ AND...AND } x_n \text{ is } \tilde{A}_n^i\\&\text{THEN } y^i \text{ is } \tilde{a}_0^i + \sum_{j=1}^{n} \tilde{a}_j^i x_j, \qquad i = 1, \ldots, M\end{aligned} \tag{1}$$

where $x_1, \ldots, x_n$ are input variables, $\tilde{A}_j^i$ represent interval fuzzy sets and $\tilde{a}_j^i$ represents interval sets, and where $\tilde{a}_j^i = [c_j^i - s_j^i, c_j^i + s_j^i]$. Hereafter, we detail the functions for each layer.

Layer 1 (Input Layer): Each node in this layer is a crisp input variable. Only input variables pass directly to layer 2, meaning that there are no weights to be adjusted in this layer.

Layer 2 (Membership Function Layer): Each node in this layer defines an interval type-2 MF to execute the fuzzification operation. For defining the interval type-2 fuzzy sets, a Gaussian primary MF, which has a fixed STD and an uncertain mean that takes values (see Fig. 2), is used

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

Each MF of the premise part can be represented as an upper, $\bar{\mu}_j^i$, and a lower MF, $\underline{\mu}_j^i$, where

$$\bar{\mu}_j^i(x_j) = \begin{cases} N\left(m_{j1}^i, \sigma_j^i; x_j\right), & x_j < m_{j1}^i \\ 1, & m_{j1}^i \le x_j \le m_{j2}^i \\ N\left(m_{j2}^i, \sigma_j^i; x_j\right), & x_j > m_{j2}^i \end{cases} \tag{3}$$

and

$$\underline{\mu}_j^i(x_j) = \begin{cases} N\left(m_{j2}^i, \sigma_j^i; x_j\right), & x_j \le \frac{m_{j1}^i + m_{j2}^i}{2} \\ N\left(m_{j1}^i, \sigma_j^i; x_j\right), & x_j > \frac{m_{j1}^i + m_{j2}^i}{2} \end{cases} \tag{4}$$

and thus, the output of this layer can be represented as an interval $[\bar{\mu}_j^i, \underline{\mu}_j^i]$.

Layer 3 (Firing Layer): Each node in this layer represents the antecedent part of a fuzzy rule and it computes the firing strength. To compute the spatial firing strength F^i, each node performs a fuzzy meet operation on the inputs that it receives from layer 2 using an algebraic product operation. The firing strength is an interval type-1 fuzzy set $[\underline{f^i}, f^i]$ and can be computed using the following:

$$F^i = \left[f^i, \underline{f^i}\right], \quad i = 1, \ldots, M \tag{5}$$

$$\underline{f^i} = \prod_{j=1}^{n} \bar{\mu}_j^i, \quad f^i = \prod_{j=1}^{n} \mu_j^i \tag{6}$$

where the index n represents the input dimension according to different examples. In Section IV-A, we explain in detail how these M rules are generated.

Layer 4 (Consequent Layer): Each node in this layer is a TSK-type consequent node and functions as a linear model with exogenous input variables. Each rule node in layer 3 has a corresponding consequent node in layer 4. The output of a consequent node is an interval type-1 set $[w_l^i, w_r^i]$ and can be expressed as

$$\left[w_l^i, w_r^i\right] = \left[c_0^i - s_0^i, c_0^i + s_0^i\right] + \sum_{j=1}^{n} \left[c_j^i - s_j^i, c_j^i + s_j^i\right] x_j. \tag{7}$$

The output of this layer can be expressed, respectively, as

$$w_l^i = \sum_{j=0}^{n} c_j^i x_j - \sum_{j=0}^{n} s_j^i |x_j| \tag{8}$$

and

$$w_r^i = \sum_{j=0}^{n} c_j^i x_j + \sum_{j=0}^{n} s_j^i |x_j| \tag{9}$$

where $x_0 \equiv 1$.

Layer 5 (Output Processing Layer): Nodes in this layer correspond to one output linguistic variable. The output function combines the output of layers 3 and 4, and the design factors $[q_l, q_r]$ enables adaptive adjustment of the upper and the lower values without using the K-M iterative procedure to find L and R end points. The use of K-M iterative procedure is the formalization, but in this case, we can use the q factors to reduce the complexity of type-2 FLS when the number of rules is larger. Thus, the output of $[y_l, y_r]$ can be computed by

$$y_l = \frac{(1 - q_l) \sum_{i=1}^{M} \underline{f^i} w_l^i + q_l \sum_{i=1}^{M} f^i w_l^i}{\sum_{i=1}^{M} \underline{f^i} + f^i} \tag{10}$$

and

$$y_r = \frac{(1 - q_r) \sum_{i=1}^{M} f^i w_r^i + q_r \sum_{i=1}^{M} \underline{f^i} w_r^i}{\sum_{i=1}^{M} \underline{f^i} + f^i}. \tag{11}$$

Layer 6 (Output Layer): The node in this layer computes the output variable using a defuzzification operation. The output of layer 5 is an interval set, and thus, node in layer 6 a sum of y_l and y_r. Finally, the defuzzified output is given by

$$y = y_l + y_r. \tag{12}$$

The above six-layer structure significantly reduces computational complexity, as will be verified through simulation results in Section IV.

IV. SIT2FNN Learning

In this section, we introduce a two-phase (structure and parameter) learning mechanism to enhance performance of SIT2FNN. The rule-base in the SIT2FNN is initially empty; all of the fuzzy rules are evolved from simultaneous structure and parameter learning as each piece of training data is received. The parameter learning phase uses a gradient descent algorithm (Section IV-B). The following sections introduce the structure and parameter learning algorithms.

A. Structure Learning

Rules are generated according to the structure learning algorithm. A previous study [4] and [40] used the rule firing strength as a criterion for type-1 fuzzy rule generation. This idea was extended to type-2 fuzzy rule generation (called an on-line type-2 fuzzy clustering) under SIT2FNN. The firing strength F^i in (5) was used to determine whether a new rule must be generated. The input space is partitioned determines the number of rules extracted from the piece of training data, as well as the number of fuzzy sets in the universe of discourse for each input variable. The type-2 rule firing strength is an interval, and the center of the spatial firing interval

$$f_c = \frac{1}{2}\left(\overline{f}^i + \underline{f}^i\right) \tag{13}$$

is used as the criterion for rule generation. The first incoming data point x is used to generate the first fuzzy rule, and the uncertain mean and width of the type-2 fuzzy MFs associated with this rule is set as

$$\left[m_{j1}^1, m_{j2}^1\right] = [x_j - \Delta x, x_j + \Delta x] \quad \sigma = \sigma_{\text{fixed}} \quad j = 1, \ldots, n \tag{14}$$

where σ_{fixed} is a predefined value ($\sigma_{\text{fixed}} = 0.3$ in this paper) that determines the width of the memberships associated with a new rule. The initial consequent parameters are set as $[c_0^1 - s_0^1, c_0^1 + s_0^1] = [y_d - 0.1, y_d + 0.1]$, where y_d is the desired output. Hereafter, for each new incoming data $x(t)$, we find

$$I = \arg\max_{1 \le i \le M(t)} f_c^i(t) \tag{15}$$

where M(t) is the number of existing rules at time t. If $f_c^I(t) \le f_{\text{th}}$ (f_{th} is a prespecified threshold), a new fuzzy rule is generated. If the present data point does not match any of the existing rules effectively, a new rule is generated. The same procedure as that for the first rule is used to assign the uncertain mean; that is, for the new rule, the uncertain mean of the corresponding type-2 fuzzy sets is defined as

$$\left[m_{j1}^{M(t)+1}, m_{j2}^{M(t)+1}\right] = \left[x_j(t) - \Delta x, x_j(t) + \Delta x\right] \quad j = 1, \ldots, n. \tag{16}$$

The width of the new rule is defined as

$$\sigma^{M(t)+1} = \beta \cdot \left| x_j - \left(\frac{m_{j1}^I + m_{j2}^I}{2} \right) \right|. \tag{17}$$

Equations (14) and (16) show that, for the mean, the width of the uncertain region is set according to different examples. If the uncertainty associated with the mean is excessively small, the type-2 fuzzy sets become similar to type-1 fuzzy sets. Conversely, if the width of the uncertain region is excessively large, the uncertain mean covers most of the input domain. Equation (17) shows that the initial width is equal to the Euclidean distance between current input data $\boldsymbol{x}$ and the center of the optimal matching rule for this data point times an overlapping parameter β. In this paper, β was set to 0.5 so that the width of the new type-2 fuzzy set was half of the Euclidean distance from the optimal matching center. This ought to achieve a suitable overlap between the adjacent rules. The newly consequent parameters must be assigned as follows:

$$c_0^{M(t)+1} = y_d(t),\ c_j^{M(t)+1} = 0.05,\ \mathrm{s}_j^{M(t)+1} = \mathrm{s}_0^{M(t)+1} = 0.002,\ j = 1, \ldots, n.$$

The entire on-line type-2 fuzzy clustering procedure for generating fuzzy rules is as follows.

On-line Type-2 Fuzzy Clustering Procedure

IF the first data $\overrightarrow{x}$ is coming THEN do

{Generate a new fuzzy rule and assign initial width and uncertain mean of each premise fuzzy set by (14)

$\left[m_{j1}^1, m_{j2}^1\right] = [x_j - 0.1, x_j + 0.1]$ and $\sigma = 0.3$, $j = 1, \ldots, n$

and also set initial consequent parameters of fuzzy rules

$c_0^1 = y_d(t), \quad c_j^1 = 0.05,\ \mathrm{s}_j^1 = \mathrm{s}_0^1 = 0.002,\ j = 1, \ldots, n$

}.

ELSE for a subsequent data $\overrightarrow{x}$ do

{Compute (15)

IF $f_c^I(t) \le f_{\text{th}}$

{Generate a new fuzzy rule $M(t+1) = M(t) + 1$ *and assign initial width and uncertain mean of new fuzzy set by* (16) and (17)

$$\left[m_{j1}^{M(t)+1},\ m_{j2}^{M(t)+1}\right] = [x_j(t) - 0.1,\ x_j(t) + 0.1]$$

$$\sigma^{M(t)+1} = \beta \cdot \left| x_j - \left(\frac{m_{j1}^I + m_{j2}^I}{2} \right) \right|$$

and also set initial consequent parameters of fuzzy rules

$c_0^{M(t)+1} = y_d(t)$

$c_j^{M(t)+1} = 0.05,\ \mathrm{s}_j^{M(t)+1} = \mathrm{s}_0^{M(t)+1} = 0.002,\ j = 1, \ldots, n$

}}.

B. Parameter Learning

In addition to the structure learning phase, all free parameters in the SIT2FNN are learned, including current parameters and those that were newly generated. For clarification, we considered the single-output case and defined the objective to minimize the error function as

$$E = \frac{1}{2}[y(t+1) - y_d(t+1)]^2 \tag{18}$$

where $y(t+1)$ and $y_d(t+1)$ are the SIT2FNN output and the desired output, respectively. According to the gradient descent algorithm, the updated parameters can be denoted by

$$c_j^i(t+1) = c_j^i(t) - \eta \frac{\partial E}{\partial c_j^i} \tag{19}$$

$$s_j^i(t+1) = s_j^i(t) - \eta \frac{\partial E}{\partial s_j^i} \tag{20}$$

$$m_{j1}^i(t+1) = m_{j1}^i(t) - \eta \frac{\partial E}{\partial m_{j1}^i} \tag{21}$$

$$m_{j2}^i(t+1) = m_{j2}^i(t) - \eta \frac{\partial E}{\partial m_{j2}^i} \tag{22}$$

$$\sigma_j^i(t+1) = \sigma_j^i(t) - \eta \frac{\partial E}{\partial \sigma_j^i} \tag{23}$$

where η is the learning rate, n is the number of input variables, and M is the number of rules, $j = 1, \ldots, n$, $i = 1, \ldots, M$. As can be observed in (24), if x_j is set to 1, then the equation updates the consequent parameter c_0^i. The derivation in (19) and (20) can be expressed by the following:

$$\begin{aligned}\frac{\partial E}{\partial c_j^i} &= \frac{\partial E}{\partial y}\left(\frac{\partial y}{\partial y_l}\frac{\partial y_l}{\partial w_l^i}\frac{\partial w_l^i}{\partial c_j^i} + \frac{\partial y}{\partial y_r}\frac{\partial y_r}{\partial w_r^i}\frac{\partial w_r^i}{c_j^i}\right)\\ &= (y-y_d)\left(\frac{(1-q_l)\bar{f}^i + q_l\underline{f}^i}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}x_j + \frac{(1-q_r)\underline{f}^i + q_r\bar{f}^i}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}x_j\right)\\ &= (y-y_d)\left(\frac{x_j}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}\right)((1-q_l+q_r)\bar{f}^i + (1-q_r+q_l)\underline{f}^i)\end{aligned} \tag{24}$$

$$\begin{aligned}\frac{\partial E}{\partial s_j^i} &= \frac{\partial E}{\partial y}\left(\frac{\partial y}{\partial y_l}\frac{\partial y_l}{\partial w_l^i}\frac{\partial w_l^i}{\partial s_j^i} + \frac{\partial y}{\partial y_r}\frac{\partial y_r}{\partial w_r^i}\frac{\partial w_r^i}{s_j^i}\right)\\ &= (y-y_d)\left(-\frac{(1-q_l)\bar{f}^i + q_l\underline{f}^i}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}|x_j| + \frac{(1-q_r)\underline{f}^i + q_r\bar{f}^i}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}|x_j|\right)\\ &= (y-y_d)\left(\frac{|x_j|}{\sum_{i=1}^{M}\bar{f}^i + \underline{f}^i}\right)((1-q_r-q_l)\underline{f}^i - (1-q_r-q_l)\bar{f}^i).\end{aligned} \tag{25}$$

The derivation of the premise part in (21)–(23) can be expressed by the following:

$$\frac{\partial E}{\partial m_{j1}^i} = (y-y_d) \times\left(\left(\frac{\partial y_l}{\partial \bar{f}^i} + \frac{\partial y_r}{\partial \bar{f}^i}\right)\frac{\partial \bar{f}^i}{\partial m_{j1}^i} + \left(\frac{\partial y_l}{\partial \underline{f}^i} + \frac{\partial y_r}{\partial \underline{f}^i}\right)\frac{\partial \underline{f}^i}{\partial m_{j1}^i}\right) \tag{26}$$

$$\frac{\partial E}{\partial m_{j2}^i} = (y-y_d) \times\left(\left(\frac{\partial y_l}{\partial \bar{f}^i} + \frac{\partial y_r}{\partial \bar{f}^i}\right)\frac{\partial \bar{f}^i}{\partial m_{j2}^i} + \left(\frac{\partial y_l}{\partial \underline{f}^i} + \frac{\partial y_r}{\partial \underline{f}^i}\right)\frac{\partial \underline{f}^i}{\partial m_{j2}^i}\right) \tag{27}$$

$$\frac{\partial E}{\partial \sigma_j^i} = (y-y_d) \times\left(\left(\frac{\partial y_l}{\partial \bar{f}^i} + \frac{\partial y_r}{\partial \bar{f}^i}\right)\frac{\partial \bar{f}^i}{\partial \sigma_j^i} + \left(\frac{\partial y_l}{\partial \underline{f}^i} + \frac{\partial y_r}{\partial \underline{f}^i}\right)\frac{\partial \underline{f}^i}{\partial \sigma_j^i}\right) \tag{28}$$

$$\frac{\partial y_l}{\partial \bar{f}^i} = \frac{(1-q_l)w_l - y_l}{\sum \bar{f} + \underline{f}}, \frac{\partial y_l}{\partial \underline{f}^i} = \frac{q_l w_l - y_l}{\sum \bar{f} + f} \tag{29}$$

$$\frac{\partial y_r}{\partial \bar{f}^i} = \frac{q_r w_r - y_r}{\sum \bar{f} + \underline{f}}, \frac{\partial y_r}{\partial \underline{f}^i} = \frac{(1-q_r)w_r - y_r}{\sum \bar{f} + \underline{f}} \tag{30}$$

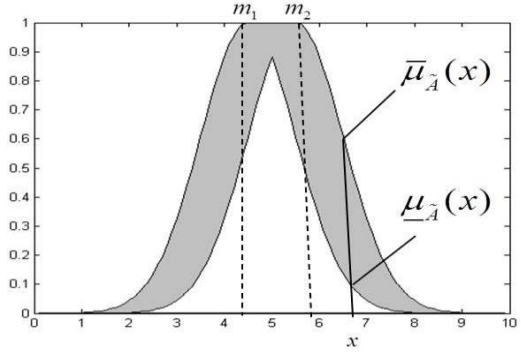

Fig. 2. Interval type-2 fuzzy set with uncertain mean.

$$\frac{\partial \bar{f}^i}{\partial m_{j1}^i} = \frac{\partial \bar{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j1}^i} = \begin{cases}\bar{f}^i \times \frac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, x_j \le m_{j1}^i\\ 0, \quad \text{otherwise}\end{cases} \tag{31}$$

$$\frac{\partial \underline{f}^i}{\partial m_{j1}^i} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j1}^i} = \begin{cases}\underline{f}^i \times \frac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, x_j > \frac{m_{j1}^i + m_{j2}^i}{2}\\ 0, \quad \text{otherwise}\end{cases} \tag{32}$$

$$\frac{\partial \bar{f}^i}{\partial m_{j2}^i} = \frac{\partial \bar{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j2}^i} = \begin{cases}\bar{f}^i \times \frac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, x_j > m_{j2}^i\\ 0, \quad \text{otherwise}\end{cases} \tag{33}$$

$$\frac{\partial \underline{f}^i}{\partial m_{j2}^i} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j2}^i} = \begin{cases}\underline{f}^i \times \frac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, x_j \le \frac{m_{j1}^i + m_{j2}^i}{2}\\ 0, \quad \text{otherwise}\end{cases} \tag{34}$$

$$\frac{\partial \bar{f}^i}{\partial \sigma_j^i} = \frac{\partial \bar{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial \sigma_j^i} = \begin{cases}\bar{f}^i \times \frac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, x_j < m_{j1}^i\\ \bar{f}^i \times \frac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, x_j < m_{j2}^i\\ 0, \quad \text{otherwise}\end{cases} \tag{35}$$

$$\frac{\partial \underline{f}^i}{\partial \sigma_j^i} = \frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \sigma_j^i} = \begin{cases}\underline{f}^i \times \frac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, x_j \le \frac{m_{j1}^i + m_{j2}^i}{2}\\ \underline{f}^i \times \frac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, x_j > \frac{m_{j1}^i + m_{j2}^i}{2}.\end{cases} \tag{36}$$

As mentioned above, the factors $[q_l, q_r]$ can be learned to adaptively adjust the upper and the lower portions of y_l and y_r in the final output. Therefore, the optimum of the factors can be derived by the following:

$$q_l(t+1) = q_l(t) - \eta\frac{\partial E}{\partial q_l} \tag{37}$$

$$q_r(t+1) = q_r(t) - \eta\frac{\partial E}{\partial q_r} \tag{38}$$

where

$$\frac{\partial E}{\partial q_l} = (y-y_d)\frac{\sum w_l(\underline{f}^i - \bar{f}^i)}{\sum \bar{f}^i + \underline{f}^i} \tag{39}$$

$$\frac{\partial E}{\partial q_r} = (y-y_d)\frac{\sum w_r(\bar{f}^i - \underline{f}^i)}{\sum \bar{f}^i + \underline{f}^i}. \tag{40}$$

For the convergence, the learning rate must be carefully selected, as a too high rate will lead to extreme oscillation in the learning process and a too low rate will result in slow learning convergence.

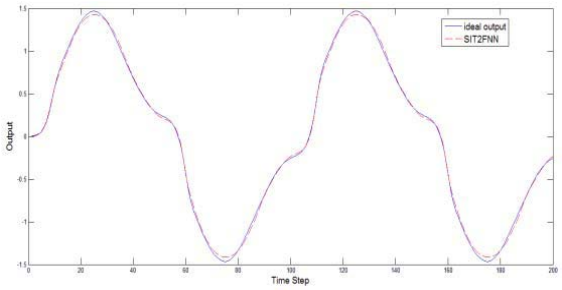

Fig. 3. Identification result of the SIT2FNN.

TABLE I
TRAINED PARAMETERS WITH THREE RULES

	Rule 1	Rule 2	Rule 3
$(m_{11}^i, m_{21}^i, \sigma_1^i)$	(-0.1202, 0.1995, 0.5838)	(0.5332, 0.9237, 0.8398)	(-0.8895, -0.4371, 0.9355)
$(m_{12}^i, m_{22}^i, \sigma_2^i)$	(-0.1969, 0.2905, 1.0777)	(0.5073, 0.8690, 0.5774)	(-0.7029, -0.4914, 0.5654)
(c_0^i, c_1^i, c_2^i)	(0.0228, -0.5466, 0.0387)	(0.7215, 0.3270, 0.1291)	(-0.5828, 0.2735, 0.1886)
(s_0^i, s_1^i, s_2^i)	(0.0036, 0.0091, 0.0367)	(0.0109, 0.00081, 0.1752)	(0.0367, 0.0126, 0.2578)

V. SIMULATION RESULT

This section presents an evaluation of the performance of the SIT2FNN through four example simulations. These simulations include two types of system identification, and two kinds of forecasting problems, including Mackey–Glass and Santa Fe time series.

A. Example 1 (System Identification)

The SIT2FNN is applied to the identification of a nonlinear system, where the data set is generated by the difference equation

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

The variables $u(t)$ and $y_d(t)$ are fed as inputs to the SIT2FNN input layer, and the desired output is $y_d(t+1)$. The training signal is generated with $u(t) = \sin(2\pi t/100)$. As a performance criterion, we evaluate the root mean square error (RMSE) as shown by the following:

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

The training procedure minimizes the square error between the output of the system $y(t+1)$ and the target output $y_d(t+1)$. As described in [4], [27], [30], [34], and [35], we follow the same computational protocols that have been used for 500 epochs with 200 samples. The structure threshold f_{th} is set to 0.01, where it influences and decides the number of fuzzy rules to be generated. The learning rate is set to 0.01. After the training process, four rules are generated and the RMSE obtained is 0.0234. Fig. 3 shows that the curve has a perfect match between the actual output of the SIT2FNN and the desired output of the system. After the whole learning process, the trained parameters are shown in Table I. As can be observed in Table II, the performance of the SIT2FNN is compared with the existing models, including an SONFIN [4], the self-adaptive FIN (SaFIN) [35], the TSK-type type-2 FNS (TSK T2FNS) [30], feedforward type-2 FNN and the SEIT2FNN. The SONFIN possesses simultaneous structure and parameter learning abilities, so for the type-1 models, it predictably outperforms SaFIN. The TSK T2 FNS just uses a design factor q to share the upper and lower values in the final output and consequent part is composed by a crisp TSK-type not an interval set. The consequent part of eT2FIS is of Mamdani-type reasoning. Like the SIT2FNN, the consequent of feedforward type-2 FNN and SEIT2FNN is of TSK-type with an interval set.

TABLE II
PERFORMANCE COMPARISON IN EXAMPLE 1

Models	No. of Rules	No. of Parameters	Test RMSE
SONFIN [4]	7	35	0.0085
SaFIN [35]	8	-	0.012
Feedforward Type-2 FNN	3	36	0.0281
T2 TSK FNS [30]	4	24	0.0324
eT2FIS[34]	14	70	0.053
SEIT2FNN[27]	3	36	0.0062
SIT2FNN	3	36	0.0241

For a meaningful comparison, the total number of parameters in type-2 models is kept similar to that of parameters in type-1 models, and thus, the number of rules used in type-1 models is larger than that in type-2 models. Type-2 models achieved consistently better performance than type-1 models. As observed in Table II, the SIT2FNN outperformed all but the SEIT2FNN. This is because the SEIT2FNN has a few performance advantages, including the ability to self-evolve ability and the use of a rule-ordered Kalman filter algorithm to tune interval weights in the consequent part. However, the SEIT2FNN will result in a larger computational cost as rule-base grows.

In general, the K-M iterative type-reduction process in type-2 systems is time-consuming. The SIT2FNN uses two design factors $[q_l, q_r]$ to replace the K-M iterative procedure and economize computational costs. Running simulations written in Visual C++ on an Intel 2.53 GHz dual central processing unit, the learning times of the feedforward type-2 FNN, SEIT2FNN, and SIT2FNN were 1.53, 1.42, and 1.00 s, respectively. These results indicate a significant performance benefit for SIT2FNN. We then compare our approach with the TSK-type T2FNS, which similarly uses a q factor to adjust the portion of the upper and lower values, and a gradient descent algorithm for learning. Again, the results indicate a clear performance advantage for SIT2FNN.

We also used simulations to assess the noise resistance capabilities of SIT2FNN, by adding artificially generated noise (uniformly distributed between $[-0.1, 0.1]$ and $[-0.5, 0.5]$) to the input signal. There are a total of 20 Monte Carlo realizations. On average, three rules are generated for the

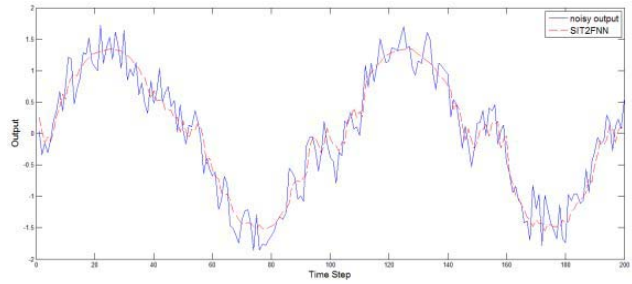

Fig. 4. One of corresponding curve of on-line learning when noise generated in the ranges [−0.5, 0.5] and appeared in the measured data.

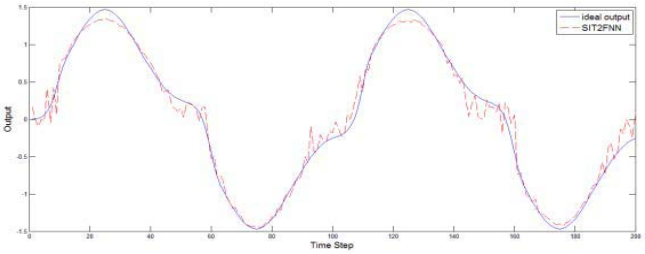

Fig. 5. One of corresponding result when measured output containing noise level in the ranges [−0.5, 0.5] for test.

TABLE III
PERFORMANCE COMPARISON WITH DIFFERENT NOISE LEVELS IN EXAMPLE 1

Models	No. of Rules	Noise	Test RMSE	No. of Rules	Noise	Test RMSE
SONFIN[4]	6	[−0.1, 0.1]	0.041	6	[−0.5, 0.5]	0.170
Feedforward Type-2 FNN	3		0.038	3		0.145
eT2FIS[34]	10		0.033	11		0.140
SEIT2FNN[27]	3		0.035	3		0.142
T2SONFS[36]	6		0.034	6		0.138
SIT2FNN	3		0.039	3		0.136

SIT2FNN. Fig. 4 shows one learning result of the 20 realizations, demonstrating that the SIT2FNN output does not vary as widely as the noisy output. Fig. 5 shows a single result when for the measured output containing noise within [−0.5, 0.5] to demonstrate superior noise tolerance of SIT2FNN. Table III shows the performance of SIT2FNN with noise. For comparison, the SONFIN [4] and the feedforward type-2 FNN were compared using the inputs. Both the SONFIN and feedforward type-2 FNN use a gradient descent algorithm to update free parameters. As can be observed in Table II, the Mamdani-type T2 models, eT2FIS [34] and T2SONFS [36], have a better performance than the TSK-type T2 models, SEIT2FNN and SIT2FNN. This is because the noise in $y(t)$ fed as inputs influences the TSK-type consequent part of each rule. Therefore, Mamdani-type fuzzy reasoning that is not a function of input variables can be less sensitive to noise than TSK-type fuzzy reasoning. For a fair comparison with TSK-type T2 models, the SIT2FNN could achieve better performance than feedforward type-2 FNN and SEIT2FNN. Obviously, type-2 models are able to effectively contend with noise conditions.

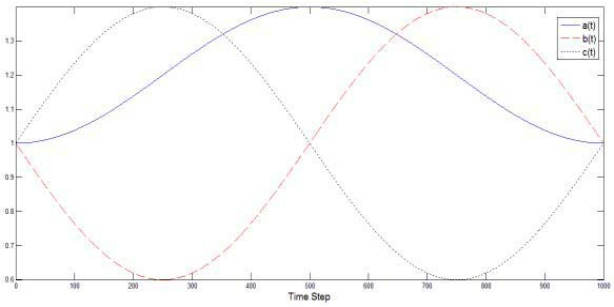

Fig. 6. Plot of corresponding time-varying signals, a(t) (solid line), b(t) (dashed line), and c(t) (dotted line).

B. Example 2 (Second-Order System Identification)

The SIT2FNN is applied to assess the performance of a second-order time-varying system. The dynamic system is introduced in [30] and is given by the following difference equation:

$$y_p(t+1) = f(y_p(t),\ y_p(t-1),\ y_p(t-2),\ u(t),\ u(t-1)) \tag{43}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - b) + c x_4}{a + x_2^2 + x_3^2} \tag{44}$$

and where the parameters, $a(t)$, $b(t)$, and $c(t)$ are expressed by

$$a(t) = 1.2 - 0.2\cos(2\pi t/T) \tag{45}$$

$$b(t) = 1.0 - 0.4\sin(2\pi t/T) \tag{46}$$

$$c(t) = 1.0 + 0.4\sin(2\pi t/T) \tag{47}$$

where T is the maximum time step. Fig. 6 shows the system signals of $a(t)$, $b(t)$, and $c(t)$.

The structure threshold and learning rate are set to be 0.2 and 0.01, respectively. After 100 epochs with 1000 time steps, four rules are generated to cover the entire input domain. To test the identification result, the following signal is used for the test:

$$u(t) = \begin{cases} \sin(\frac{\pi t}{25}), & t < 250 \\ 1.0, & 250 \le t < 500 \\ -1.0, & 500 \le t < 750 \\ 0.3\sin(\frac{\pi t}{25}) + 0.1\sin(\frac{\pi t}{32}) & \\ \quad +0.6\sin(\frac{\pi t}{10}), & 750 \le t < 1000. \end{cases} \tag{48}$$

Fig. 7 shows the identification result of SIT2FNN. Table IV shows the performance of SIT2FNN, including the number of rules and epochs as well as the training and test RMSEs. Apparently, type-2 models use fewer rules compared with type-1 TSK FNS, but their performance can achieve the lower RMSE. The consequent part of the compared type-2 models except for the T2 TSK FNS consists of TSK-type with interval sets. When compared with T2 TSK FNS, SIT2FNN, which also uses a gradient descent algorithm and design factors $[q_l, q_r]$ (to replace the K-M iterative procedure), achieves notably better performance and a smaller RMSE than all other type-2 models. As in Example 1, we also consider the execution time of training process among feedforward type-2

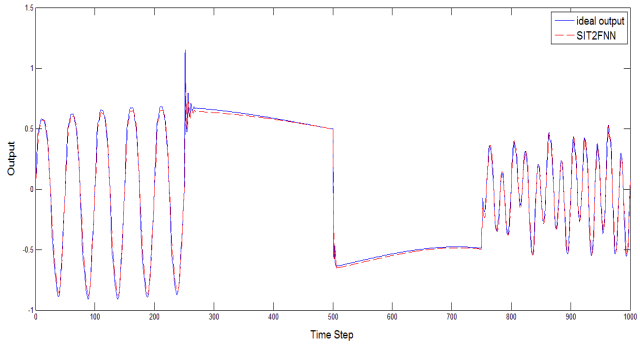

Fig. 7. Simulation result between ideal output (solid line) and SIT2FNN (dashed line).

TABLE IV
PERFORMANCE COMPARISON IN EXAMPLE 2

Models	No. of Rules	No. of Parameters	Training RMSE	Test RMSE
T1 TSK FNS [30]	9	63	0.0282	0.0598
T2 TSK FNS [30]	4	24	0.0284	0.0601
Feedforward Type-2 FNN	4	36	0.0339	0.0587
SEIT2FNN [27]	4	36	0.0344	0.0577
SIT2FNN	4	36	0.0351	0.0560

FNN, SEIT2FNN, and SIT2FNN, and their execution times take 2.52, 2.55, and 0.9 s, respectively. The training process of SIT2FNN is much faster than its rivals.

C. Example 3 (Noisy Chaotic Time Series Prediction)

This example is meant to predict the Mackey Glass time series, which is generated using the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{49}$$

where $\tau \geq 17$. As in [4], [27], [31], and [34], the systems response was chaotic, and simulation data were obtained using initial conditions $x(0) = 1.2$ and $\tau = 30$. Four past values are used to predict $x(t)$, and the input–output data format is $[x(t-24), x(t-18), x(t-12), x(t-6); x(t)]$. A total of 1000 data pairs were obtained from the interval $t \in [124, 1123]$. The first 500 patterns are used for training and the remaining 500 for testing. The structure threshold and learning coefficient are set to 0.2 and 0.07, respectively. After 500 epochs of training process, seven rules are generated. For noise-free conditions, the training and test RMSEs of SIT2FNN are 0.00479 and 0.00722, respectively, again demonstrating costs of the K-M iterative procedure in type-2 models.

As in Example 1, we also investigated the computational advantage of SIT2FNN, and found that the learning time for SIT2FNN was 6.0 s, compared with 93.4 s (15.5 times slower) for SEIT2FNN and 37.3 s (6.21times slower) for feedforward type-2 FNN. Again, the proposed model significantly reduces computational cost as rule-base is larger.

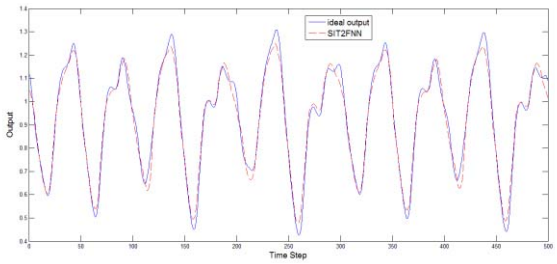

Fig. 8. Prediction results of SIT2FNN, trained with noise level STD = 0.1 and noise-free for test.

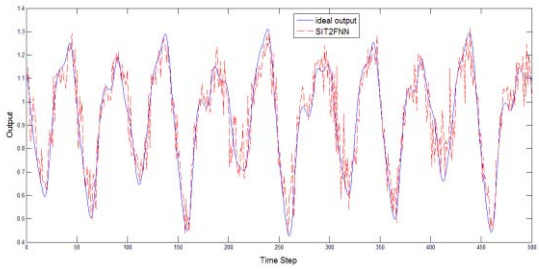

Fig. 9. Prediction results of SIT2FNN, trained with noise level STD = 0.1 and noise level STD = 0.1 for test.

TABLE V
PERFORMANCE COMPARISON WITH NOISE LEVEL STD = 0.1 IN EXAMPLE 3

Models	SONFIN [4]	SEIT2FNN [27]	IT2FNN-SVR [31]	eT2FIS [34]	SIT2FNN
No. of Rules	10	5	6	--	5
No. of Parameters	130	110	103	--	110
Training RMSE (STD=0.1)	0.113	0.123	0.127	0.120	0.088
Test RMSE (Clean)	0.054	0.049	0.046	0.059	0.041
Test RMSE (STD=0.1)	0.108	0.097	0.088	0.107	0.087
Test RMSE (STD=0.3)	0.256	0.212	0.215	0.214	0.215

The SIT2FNN has also shown more robust with respect to measurement noise as shown below. Assume two levels of Gaussian noise with STDs 0.1 and 0.3 are added to $x(t)$ for training, and a total of 10 Monte Carlo realizations for the statistical analysis of the results. Figs. 8 and 9 show the prediction result of the SIT2FNN for test in terms of noise-free and noise level STD = 0.1. Tables V and VI together show the performance of SIT2FNN for different noise environments, including training for STD = 0.1 with respect to the test for noise-free, STD = 0.1 and STD = 0.3, respectively; and training for STD = 0.3 with respect to the test for noise-free, STD = 0.1 and STD = 0.3, respectively. For the purpose of comparison, we also use the same noisy environment to assess the noise tolerance of the other networks, including the SONFIN and SEIT2FNN, IT2FNN-SVR [31], and eT2FIS [34]. Obviously, the type-2 models, namely SEIT2FNN, IT2FNN-SVR, eT2FIS, and SIT2FNN, are superior to the SONFIN

TABLE VI
PERFORMANCE COMPARISON WITH NOISE LEVEL STD = 0.3 IN EXAMPLE 3

Models	SONFIN [4]	SEIT2FNN [27]	IT2FNN-SVR [31]	eT2FIS [34]	SIT2FNN
No. of Rules	10	5	6	--	5
No. of Parameters	130	110	103	--	110
Training RMSE (STD=0.3)	0.302	0.319	0.347	0.327	0.166
Test RMSE (Clean)	0.195	0.196	0.121	0.102	0.121
Test RMSE (STD=0.1)	0.208	0.197	0.131	0.152	0.130
Test RMSE (STD=0.3)	0.305	0.254	0.184	0.278	0.176

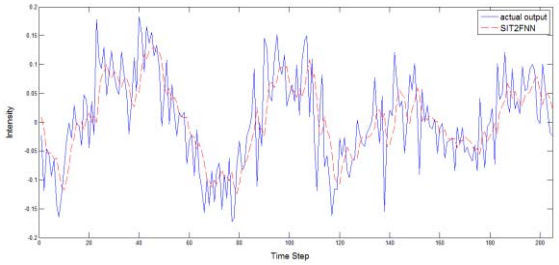

Fig. 10. Prediction results of the SIT2FNN for the Series-E of Santa Fe data set.

TABLE VII
PERFORMANCE COMPARISON IN EXAMPLE 4

Models	No. of Rules	No. of Parameters	Test RMSE
NN	--	--	0.279
ES	--	--	0.182
PMRS [39]	--	--	0.123
Feedforward type-1 FNN	12	96	0.064
Feedforward type-2 FNN	7	105	0.069
SIT2FNN	3	45	0.056

when training data contains the higher noise level with STD = 0.3.

As can be observed in Tables V and VI, the performance of IT2FNN-SVR is similar to that of SIT2FNN, but its computational is quite intensive. In other words, the SIT2FNN can achieve similar performance with less computational complexity. These results indicate type-2 sets possess considerably better noise tolerance compared than type-1 sets. In comparison with K-M iterative procedure, the IT2FNN-SVR seems to be approximately better than other type-2 models. Finally, the SIT2FNN achieves the best performance over the other compared FNNs for noise-free and noisy cases.

D. Example 4 (Practical Time Series Prediction)

In this paper, we conduct to assess the performance of SIT2FNN for a real-world series database [38] that is the Santa Fe Time series downloaded from the website www.psych.stanford.edu/~andreas/Time-Series/. This is a set of measurements of the light curve (time variation of the intensity) of the variable white dwarf star PG1159-035 and it is very noisy and discontinuous. As in [29] and [39], the aim is to predict the intensity of the star $y_p(t+1)$ based on its past intensities. We only use the intensity $y_p(t)$ as input to the SIT2FNN input layer. For a total of 2048 observations, we use 90% and 10% of the total samples for training and test process. The structure threshold and learning rate are set to 0.3 and 0.01, respectively. After 100 epochs of training, there are three rules generated to cover the entire input domain. Fig. 10 shows the actual and SIT2FNN predicted intensities. As can be observed in Table VI, the performance of SIT2FNN is compared with that of NN, PMRS, ES, and feedforward type-1 and type-2 FNN. All comparison models use the same input variables but for NN. The NN uses the past five intensities as inputs. The PMRS [39], namely pattern modeling and recognition system, was reported for addressing the same series. The ES, namely statistical exponential smoothing, was also reported in this paper. The numbers of rules in the feedforward type-1 and type-2 FNN are 12 and 7, respectively. The parameter learning of feedforward type-1 and type-2 FNN is based on a gradient descent algorithm. It seems that Type-1 models for addressing considerable noise are insufficient compared with the SIT2FNN. The computational cost in the SIT2FNN takes 0.86 s with rule number equal to three, and feedforward type-2 FNN and SIT2FNN, respectively, takes 51.0 and 1.65 s when number of rules is equal to seven. Apparently, our approach is able to vastly reduce computational complexity. Cleary from Table VII, the SIT2FNN with a fewer rules could yield smaller error than its competitors.

VI. CONCLUSION

In this paper, a simplified TSK-type IT2FNN with simultaneous structure and parameter learning online is proposed. The premise clause of type-2 fuzzy sets is shown to be sufficient to address numerical uncertainty as well as uncertain information. The structure learning algorithm enables the network to efficiently generate the required network structure, obviating the need for any initial network structure. All free updated weights are derived using a gradient descent algorithm. This is especially useful for dealing with problems with time-varying characteristics. Using factors $[q_l, q_r]$ to replace K-M iterative procedure, the SIT2FNN can significantly reduce computational cost. Simulations of identification and practical time series prediction showed that the SIT2FNN outperforms competing FNNs under a variety of realistic conditions.

REFERENCES

[1] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1997, ch. 17.
[2] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, Dec. 1991.

[3] 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.

[4] 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. 12–31, Feb. 1998.

[5] K. C. Tan and H. J. Tang, "New dynamical optimal learning for linear multilayer FNN," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1562–1570, Nov. 2004.

[6] C. F. Hsu, "Self-Organizing Adaptive fuzzy neural control for a class of nonlinear systems," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 1232–1241, Jul. 2010.

[7] S. Yilmax and Y. Oysal, "Fuzzy wavelet neural network models for prediction and identification of dynamical systems," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1599–1609, Oct. 2010.

[8] D. Wang, X. J. Zeng, and J. A. Keane, "A simplified structure evolving method for Mamdani fuzzy system identification and its application to high-dimensional problems," *Inf. Sci.*, vol. 220, no. 20, pp. 110–123, Jan. 2013.

[9] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975.

[10] 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.

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

[12] 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.

[13] C. Wagner and H. Hagras, "Toward general type-2 fuzzy logic systems based on zSlices," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 637–660, Apr. 2010.

[14] D. Wu and J. M. Mendel, "Computing with words for hierarchical decision making applied to evaluating a weapon system," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 441–460, Apr. 2010.

[15] J. M. Mendel, "Comments on 'α-plane representation for type-2 fuzzy sets: Theory and applications,"' *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 1, pp. 229–230, Feb. 2010.

[16] X. X. Zhang, H. X. Li, and C. K. Qi, "Spatially constrained fuzzy-clustering-based sensor placement for spatiotemporal fuzzy-control system," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 946–957, Oct. 2010.

[17] A. Niewiadomski, "On finity, countability, cardinalities, and cylindric extensions of type-2 fuzzy sets in linguistic summarization of databases," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 532–545, Jun. 2010.

[18] 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.

[19] N. N. Karnik and J. M. Mendel, "Application of type-2 fuzzy logic systems to forecasting of time-series," *Inf. Sci.*, vol. 120, nos. 1–4, pp. 89–111, Nov. 1999.

[20] 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.

[21] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak, "Analysis of the noise reduction property of type-2 fuzzy logic system using a novel type-2 membership function," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1395–1406, Oct. 2011.

[22] 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.

[23] C. H. Lee, J. L. Hong, Y. C. Lin, and W. Y. Lai, "Type-2 fuzzy neural network systems and learning," *Int. J. Comput. Cognit.*, vol. 1, no. 4, pp. 79–90, Dec. 2003.

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

[25] C. H. Wang, C. S. Cheng, and T. T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 1462–1477, Jun. 2004.

[26] 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.

[27] C. F. Juang and Y. W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, Dec. 2008.

[28] J. Y. Chang, Y. Y. Lin, M. F. Han, and C. T. Lin, "A functional-link based interval type-2 compensatory fuzzy neural network for nonlinear system modeling," in *Proc. Int. Conf. Fuzzy Syst.*, Jun. 2011, pp. 939–943.

[29] C. F. Juang, R. B. Huang, and Y. Y. Lin, "A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1092–1105, Oct. 2009.

[30] R. H. Abiyev and O. Kaynak, "Type 2 fuzzy neural structure for identification and control of time-varying plants," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4147–4159, Dec. 2010.

[31] C. F. Juang, R. B. Huang, and W. Y. Cheng, "An interval type-2 fuzzy neural network with support vector regression for noisy regression problems," *IEEE Trans. Fuzzy syst.*, vol. 18, no. 4, pp. 686–699, Aug. 2010.

[32] O. Uncu and I. B. Truksen, "Discrete interval type 2 fuzzy system models using uncertainty in learning parameters," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 90–106, Feb. 2007.

[33] C. H. Lee, T. W. Hu, C. T. Lee, and Y. C. Lee, "A recurrent interval type-2 fuzzy neural network with asymmetric membership functions for nonlinear system identification," in *Proc. Int. Conf. Fuzzy Syst.*, Jun. 2008, pp. 1496–1502.

[34] S. W. Tung, C. Quek, and C. Guan, "eT2FIS: An evolving type-2 neural fuzzy inference system," *Inf. Sci.*, vol. 220, no. 20, pp. 124–148, Jan. 2013.

[35] S. W. Tung, C. Quek, and C. Guan, "SaFIN: A self-adaptive fuzzy inference network," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1928–1940, Dec. 2011.

[36] C. F. Juang and Y. W. Tsao, "A type-2 self-organizing neural fuzzy system and its FPGA implementation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1537–1548, Dec. 2008.

[37] Y. Takahashi, "Adaptive predictive control of nonlinear time varying systems using neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar./Apr. 1993, pp. 1464–1468.

[38] A. S. Weigend and N. A. Gersehnfield, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA, USA: Addison-Wesley, 1994.

[39] S. Singh, "Noise impact on time-series forecasting using an intelligent pattern matching technique," *Pattern Recognit.*, vol. 32, no. 8, pp. 1389–1398, Aug. 1999.

[40] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 310–321, Feb. 2013.

[41] W. L. Tung and C. Quek, "GenSoFNN: A generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1075–1086, Sep. 2002.

[42] P. Liu and H. Li, "Efficient learning algorithms for three-layer regular feedforward fuzzy neural networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 545–558, May 2004.

[43] J. R. Castro, O. Castillo, P. Melin, and A. Rodríguez Díaz, "A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks," *Inf. Sci.*, vol. 179, no. 13, pp. 2175–2193, 2009.

[44] J. R. Castro, O. Castillo, P. Melin, O. Mendoza, and A. Rodríguez Díaz, "An interval type-2 fuzzy neural network for chaotic time series prediction with cross-validation and akaike test," *Studies Comput. Intell.*, vol. 318, pp. 269–285, 2011.

[45] O. Castillo and P. Melin, "Optimization of type-2 fuzzy systems based on bio-inspired methods: A concise review," *Inf. Sci.*, vol. 205, pp. 1–19, Nov. 2012.

[46] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers," *Eng. Appl. Artif. Intell.*, vol. 19, no. 8, pp. 829–841, Dec. 2006.

[47] D. Hidalgo, P. Melin, and O. Castillo, "An optimization method for designing type-2 fuzzy inference systems based on the footprint of uncertainty using genetic algorithm," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4590–4598, 2012.

[48] Y. Maldonado, O. Castillo, and P. Melin, "Particle swarm optimization of interval type-2 fuzzy systems for FPGA applications," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 496–508, 2013.

[49] C. F. Juang and C. H. Hsu, "Reinforcement interval type-2 fuzzy controller design by online rule generation and Q-value-aided ant colony optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1528–1542, Dec. 2009.

[50] M. Biglarbegian, W. W. Melek, and J. M. Mendel, "On the stability of interval type-2 TSK fuzzy Logic Control Systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 3, pp. 798–818, Jun. 2010.

[51] B. Gabry and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 769–783, May 2000.

[52] S. F. Su and F.-Y. P. Yang, "On the dynamical modeling with neural fuzzy networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1548–1553, Nov. 2002.
[53] C. F. Juang, C. T. Chiou, and C. L. Lai, "Hierarchical singleton-type Recurrent neural networks for noisy speech recognition," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 833–843, May 2007.
[54] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "A TSK-type based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 447–459, Jan. 2014.

Yang-Yin Lin received the M.S. degree from the Institute of Electrical Engineering, National Chung-Hsing University, Taichung, Taiwan, in 2008, and the Ph.D. degree from the Department of Electrical Engineering, National Chiao-Tung University, Hsinchu, Taiwan, in 2013.

He is currently with the Brain Research Center, National Chiao-Tung University, as a Post-Doctoral Researcher. His current research interests include computational intelligence, type-2 fuzzy systems, neural networks, and FPGA chip design.

Shih-Hui Liao was born in Tainan, Taiwan, in 1983. She received the B.S. degree from the Department of Mechatronics Engineering, Changhua University of Education, Changhua, Taiwan, in 2007, and the M.S. degree from the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2009. She is currently pursuing the Ph.D. degree in electrical and control engineering with National Chiao-Tung University, Hsinchu, Taiwan.

Her current research interests include machine learning, soft computing, and evolutionary algorithm.

Jyh-Yeong Chang (S'84–M'86) received the B.S. degree in control engineering in 1976 and the M.S. degree in electronic engineering in 1980 from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, 1976 and 1980, respectively, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 1987.

He was a Research Fellow with the Chung Shan Institute of Science and Technology, Lung-Tan, Taiwan, from 1976 to 1978 and 1980 to 1982. In 1987, he was an Associate Professor with the Department of Electrical and Control Engineering, NCTU, where he is currently a Professor. His current research interests include neural fuzzy systems, video processing, surveillance, and bioinformatics.

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the B.S. degree from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively.

He is currently the Provost and Chair Professor of electrical and computer engineering, and the Director of the Brain Research Center, NCTU. He has published more than 120 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuro-engineering, including approximately 74 IEEE journal papers. He is the co-author of *Neural Fuzzy Systems* (Prentice-Hall) and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning*.

Dr. Lin is an Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He served on the Board of Governors at the IEEE Circuits and Systems Society from 2005 to 2008, the IEEE Systems, Man, Cybernetics Society from 2003 to 2005, the IEEE Computational Intelligence Society from 2008 to 2010, and was a Chair of the IEEE Taipei Section from 2009 to 2010. He served as the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II from 2006 to 2008.