An Interval Type-2 Neural Fuzzy System for online system identification and Feature Elimination (IT2NFS-SIFE)

Yang-Yin Lin, Nikhil R. Pal, *Fellow, IEEE* and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*— We propose an integrated mechanism for discarding derogatory features and extraction of fuzzy rules based on an interval type-2 neural fuzzy system – in fact, it is a more general scheme that can discard bad features, irrelevant antecedent clauses, and irrelevant rules. High-dimensional input and large number of rules, enhance the computational complexity of neural fuzzy systems (NFSs) and reduce their interpretability. So a mechanism for simultaneous extraction of fuzzy rules and reducing the impact of (or eliminating) the inferior features is necessary. The proposed IT2NFS-SIFE uses type-2 fuzzy sets to model uncertainties associated with information and data in designing the knowledge base. The consequent part of the IT2NFS-SIFE is of Takagi-Sugeno-Kang (TSK) type with interval weights. The IT2NFS-SIFE possesses self-evolving property that can automatically generate fuzzy rules as and when required and discard poor features and antecedent clauses based on the concept of a membership modulator. The antecedent and modulator weights are learned using a gradient descent algorithm. The consequent part weights are turned via the rule-ordered Kalman filter algorithm to enhance learning effectiveness. Simulation results exhibit that IT2NFS-SIFE not only simplifies the system architecture by eliminating derogatory/irrelevant antecedent clauses, rules and features but also can maintain excellent performance.**

***Index Terms*— type-2 fuzzy neural networks, on-line type-2 fuzzy clustering, fuzzy identification, feature selection.**

I. Introduction

RECENTLY, a considerable number of type-2 fuzzy logic systems (T2FLS) to contend uncertainties associated with fuzzy rule bases are proposed in [1]–[10]. When the circumstances are uncertain (data are corrupted by noise) to determine exact membership grades, T2FLS is a very important tool to effectively address this problem. Like a type-1 FLS, a type2 FLS too involves fuzzifier, rule base, fuzzy inference engine, and output processor. For acquiring a type-1 fuzzy output and crisp values, in case of T2FLS, the output processor involves a type-reducer and a defuzzifier. As a result, many researchers have reported [11]–[15] to construct premise parts of fuzzy rule clause using type-2 fuzzy sets. However, for a general type-2 fuzzy system, it is difficult to use type-reduction, which is computationally intensive. Consequently, Type-2 fuzzy systems generally use interval type-2 fuzzy sets. Many methods have been proposed to design of interval type-2 neural fuzzy systems [11], [12], [16]–[30] (IT2NFS) having the capability of fuzzy reasoning to handle uncertain situation and learning abilities like that of neural networks. The IT2NFSs have been successfully applied in many applications including control [19], [29], identification [11], [16]–[18], bio-engineering [31], temperature prediction [62], [63] and time series forecasting [20], [22]. In [17], the self-evolving interval type-2 NFS is proposed for addressing nonlinear time-varying plants, where it uses hybrid learning algorithms - a gradient descent scheme to tune the premise part parameters and the rule-ordered Kalman filter to tune the consequent part parameters. In [29], the authors propose a type-2 fuzzy neural structure (T2 FNS) for identification and control of time-varying plants, where the constructed structure is based on type-2 Takagi-Sugeno-Kang (TSK) fuzzy rules, whose antecedent parts are derived by interval type-2 fuzzy sets and the consequent parts use crisp linear functions.

The tasks of identification of the FNN as well as realization of good performance by the identified system are often significantly dependent on the input variables. If the input is in high dimension, the identification of useful fuzzy systems becomes very difficult. Moreover, in high dimension, it loses its interpretability by human beings, where interpretability is one of the main attractions of a fuzzy system. It is also known that more inputs not necessarily favor the system identification task if there are useless features as well as redundant features. For a fuzzy neural network, a few redundant features may result in a higher computational cost and lead to more difficulty in identifying the system. Therefore, for high-dimensional inputs, use of the useful features as done in [32]–[37] is always desirable and it could reduce the computational complexity. In [33], the authors have used a measure of fuzzy entropy as the criterion to select useful features in conjunction with the backward elimination method. On the other hand, feature selection using mutual information for fuzzy random variables is presented in [36]. According to the authors of [36], use of features selected by mutual information for fuzzy rule based system may be desirable because in a fuzzy rule base we use fuzzified variables. Hence they defined and used mutual information between two fuzzified random variables. In [35], authors proposed a method that quantifies the discriminative power of input features on a fuzzy model and this is used for feature relation in connection with designing fuzzy models. For feature selection and fuzzy rule extraction, several studies [31], [34], [38]–[41] have used neural network-based methods. In,

[31], [38], [39] the authors proposed use of feature modulators which are tuned through the process of learning to select useful features in an integrated manner simultaneously with designing of the decision making system. The modulator functions associated with the antecedent clauses play significantly roles in selection of good features (removing the bad features and associated antecedent clauses). An integrated mechanism to select useful feature is also proposed in [42] which designs the rule base for function approximation type problem in the Takagi-Sugeno framework. Some of these studies [31], [38], [39], [43], [44] using layered architectures have proposed integrated mechanisms for feature selection and rule extraction, which can exploit subtle nonlinear interaction between features and that between features and the fuzzy rule base to identify useful features and hence rules.

All of aforementioned models and methods are for type-1 fuzzy systems. To the best of our knowledge, there is no type-2 fuzzy rule based system or neuro-fuzzy system for simultaneous feature selection and system identification. These methods either reject a feature or accept it. These methods cannot deal with situations when a particular feature is useful for some rules but not for all. This is what we develop here. We develop a feature modulator based on a self-evolving interval type-2 NFS with simultaneous feature selection and rule extraction schemes. The proposed method has several distinguished characteristics as follows: (i) the self-evolving property in the IT2NFS-SIFE can automatically evolve the required network structure and parameters based on the training data; (ii) the proposed IT2NFS-SIFE enables us to eliminate derogatory or harmful features to economize computational cost. In other words, our proposed network can reject truly bad features and simplify fuzzy rules); (iii) it can identify partially useful features that are important for some rules but not for all; (iv) for the consequent updated parameters, the IT2NFS-SIFE uses the rule-ordered Kalman filter algorithm to enhance network's performance; and (v) the convergence of the IT2NFS-SIFE is faster, and it achieves a lower RMSE than other models, as shown in Section V.

The concept of modulator based feature selection for Type-1 systems is well developed. But there are a few distinct differences. All type-1 approaches either select a feature or discard a feature. Therefore, every rule uses the same set of linguistic (input) variables. The proposed system, is a further generalization of the concept. Here we have a separate modulator for every atomic antecedent clause. Therefore, depending on the rule, the number of input variables involved could be different. The Type-1 approaches can only eliminate poor features, but the present approach can also eliminate useless rules. Moreover, this is the first attempt to use such a concept for Type-2 systems. Many studies [35, 36, 40] used two phases to build a whole framework in terms of feature selection to find useful attributes and then designing neural fuzzy networks as classifiers. The other studies [50, 58] used the on-line self-organizing FNN with growing and pruning strategy to enable the network to optimize the size of the FNN. In this paper, we present a self-evolving approach that embeds a pruning strategy into the fuzzy rule generation process for function approximation. Unlike the mentioned methods, our proposed method employs a simple function as an attenuating gate to probe which features may affect the system's performance adversely to discard them. The method not only speeds up the structure learning process but also removes less-useful and invalid weights and rules to achieve a parsimonious fuzzy neural network which can obtain comparable performance and accuracy. In Section II, we described the pros and cons of various approaches.

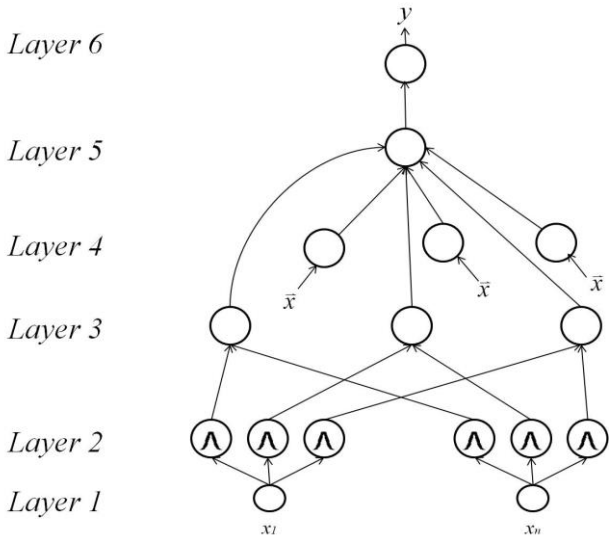

Fig. 1. Structure of the IT2NFS-SIFE

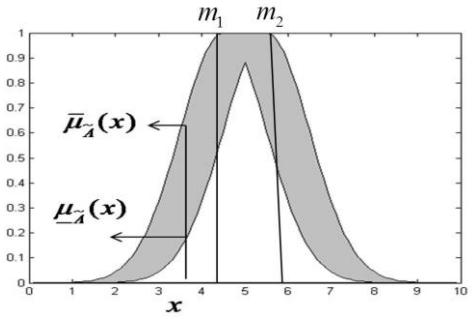

Fig. 2. An interval type-2 fuzzy set of Gaussian shape with uncertain means

The remainder of this paper is organized as follows: Section II illustrates brief survey of some existing methods; Section III introduces the IT2NFS-SIFE structure; Section IV presents the structure and parameter learning in the IT2NFS-SIFE; Section V outlines the simulation results obtained from the different cases; and lastly, Section VI offers a conclusion.

II. Brief Survey of Some Existing Methods

In real world problems, the obtained information is often uncertain and imprecise and hence it is difficult to analyze using type-1 FLS. The T2FLS was developed to deal with such problems. Practical applications of type-2 fuzzy logic systems have drawn much attention in a wide variety fields such as control, bio-medical application, temperature prediction and signal processing [11, 16 - 19, 29, 31, 62, 63]. Since a general type-2 fuzzy neural network (FNN) is computationally

intensive, the IT2FLS, which uses interval type-2 fuzzy sets in the premise part of fuzzy rules, are developed to simplify the computational complexity of general type-2 fuzzy systems. Afterwards, considerable research has been devoted to develop combinations of T2FLS and NN (named T2NFS) [11], [16]–[19], [27], [29]. In [21], [23] and the dynamic optimal training for the two-layer consequent part of interval T2FNN with fixed fuzzy rules. To yield a better performance, genetic algorithms (GAs) are used to search for the global optimal solutions of the relevant weights in terms of the premise and consequent fuzzy clauses. In some literatures [45]–[49], GAs are used to evolve the number of rules required. However, a larger number of rules result in a long complex chromosome and demand high execution time.

In [29], the structure of a type-2 TSK fuzzy neural system was presented, and its structure identification was done using a type-2 type- fuzzy c-means (FCM) algorithm, which is an extended version of the type-1 FCM. The use of an FCM type clustering algorithm demands that data must be collected in advance, and thus, it is not suitable for addressing dynamic system modeling. There are many studies [16], [17], [27], [44], [50]–[52] where researchers have advocated on-line structure identification and parameter learning approaches in order to effectively deal with the problems of time-varying system. One of well-known on-line learning networks is a self-evolving interval type-2 fuzzy neural network (SEIT2FNN) [17] with concurrent structure and parameter learning. Apart from the natural benefits of using type-2 FLS, other advantages of SEIT2FNN come from two sources : the rule adaptation mechanism that generates rules online as the system receives the training data and the rule-ordered Kalman filter algorithm that enhances network's performance. Consequently, considerable research effort has been focused on developing a self-organizing protocol that is superior for real-life application. Although the aforementioned T2FNNs possesses a distinguished characteristic that could address uncertainties associated with data and information in the knowledge base of the process very well, it still has no mechanism that can automatically discard derogatory features as the learning proceeds.

Features could be classified four groups [53]: 1) essential features, which are necessary regardless of the modeling tools used; 2) derogatory features, which must be dropped; 3) indifferent features, which are useless and neither help nor cause problems except possibly increasing cost; and 4) redundant features, which are useful but all of them are not needed because of dependency. Feature selection is important not only to reduce the cost of design and decision making but also to make the learning efficient and easy. In [54]–[56], the authors, who made important contributions in feature selection, dealt with a combinatorial optimization problem with two conflicting objectives: minimization of the number of rules and maximization of the accuracy. In [36], the authors presented a novel feature selection scheme using mutual information for fuzzy random variables that could prevent poor features to impact the performance of the system. However, these methods perform feature selection in an offline manner for the classification task. To tackle this problem, several methods of integrated feature selection and rule extraction are proposed using layered networks [31], [38], [42], [50], [57], [58]. The demerits of some of the proposed networks [31], [38], [42] include conflicting rules that need to be eliminated by post-processing. In [57], the authors presented an integrated mechanism that can account for possible subtle nonlinear interaction between features for simultaneous extraction of fuzzy rules and selection of useful features. However, that approach only discards harmful features but cannot simplify the network's size. In [50] a novel growing and pruning mechanism is proposed, which optimizes the structure of a fuzzy neural network. The structure identification relies on sensitivity analysis. These models do not consider issues like feature selection to reduce input dimension and rule adaptation. Reducing the numbers of useless fuzzy rules and/or insignificant/poor features, which enhances the interpretability of the system, usually has a positive impact on the performance of system. Here, the objectives of this study are to select useful membership functions (antecedent clauses) and features and to eliminate derogatory features and not-useful membership functions through a modulation operation that could easily identify good/bad features during the leaning process, and then use the significant features to achieve better system performance.

III. IT2NFS-SIFE STRUCTURE

Here we introduce the structure of the multiple-input-single-output IT2NFS-SIFE. The antecedent part of the IT2NFS-SIFE uses interval type-2 fuzzy sets with uncertain means and fixed standard derivation (STD). The consequent part of the IT2NFS-SIFE is of Takagi-Sugeno-Kang (TSK) type with interval weights. Fig. 1 shows the proposed five-layered IT2NFS-SIFE structure. The details of each layer are discussed next.

Layer 1 (*Input Layer*): Each node in this layer represents a crisp input variable. The input variables $\vec{x} = (x_1, \cdots, x_n)$ are fed into this layer. Since this is a fan-out layer, there is no weight to be adjusted in this layer.

Layer 2 (*Modulated Membership Function Layer*): Each node in this layer performs the fuzzification task and it also modulates memberships depending on the utility of the associated feature to solve the problem. First, we use a Gaussian primary MF with a fixed STD and uncertain mean that acquires values in $[m^i_1, m^i_2]$ (see Fig. 2). Thus given a crisp value of the input variable, the output of the Gaussian membership function, $\tilde{A}^i_j$, can be represented as an interval $[\underline{\mu}^i_j, \overline{\mu}^i_j]$

$$\mu_{\tilde{A}^i_j} = \exp\{-\frac{1}{2}\left(\frac{x_j - m^i_j}{\sigma^i_j}\right)^2\} \equiv N(m^i_j, \sigma^i_j; x_j) \quad (1)$$

Here $\tilde{A}^i_j$ is the ith interval type-2 fuzzy set on the input variable $x_j, j = 1, \cdots, n$. The footprint of uncertainty (FOU) of this MF is represented by the bounded area defined in terms of a lower MF, $\underline{\mu}^i_j$, and an upper MF, $\overline{\mu}^i_j$, where

$$\overline{\mu}_{\tilde{A}^i_j}(x_j) = \begin{cases} N(m^i_{j1}, \sigma^i_j; x_j), & x_j < m^i_{j1} \\ 1, & m^i_{j1} \le x_j \le m^i_{j2} \\ N(m^i_{j2}, \sigma^i_j; x_j), & x_j > m^i_{j2} \end{cases} \quad (2)$$

and

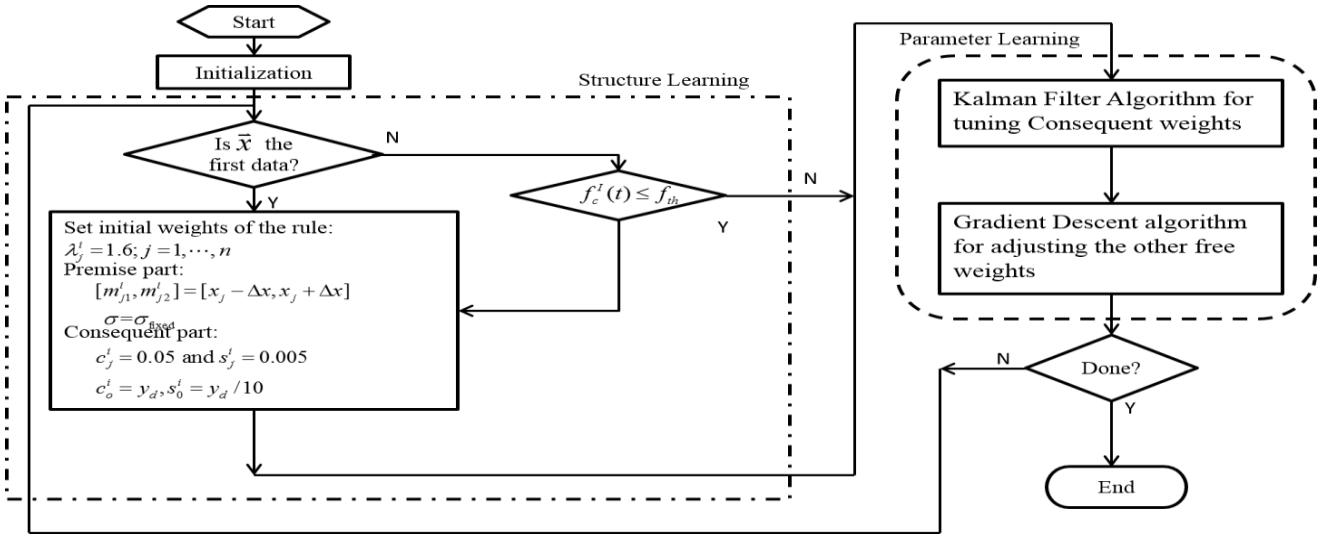

Fig. 3. Flowchart of the structure and parameter learning of the IT2NFS-SIFE

$$\underline{\mu}_{\tilde{A}_j^i}(x_j) = \begin{cases} N(m_{j2}^i, \sigma_j^i; x_j), & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ N(m_{j1}^i, \sigma_j^i; x_j), & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \end{cases} \quad (3)$$

For feature selection, our purpose is to modulate the MF associated with features in this layer. To realize this, we combine a modulator function and the Gaussian membership function in a judicious manner. For a harmful feature, the interval type-2 membership should be changed in such a manner that it does not influence the firing interval that will be computed in the next layer. However, there could be features which are useful for some rules and not for some other rules. To account for this, unlike previous methods here we have separate modulators for each membership function (linguistic value, in other words with each antecedent clause). We shall explain how to use the modulator in this layer after we look at how we compute the firing strength in the next layer.

Layer 3 (Firing Strength Layer): Each node in this layer represents the antecedent part of a fuzzy rule and it computes the firing strength. To acquire the spatial firing strength F^i, each node performs a fuzzy meet operation on inputs that it receive from layer 2 using an algebraic product operation. In this layer, there are M nodes, each of which represents one rule. The firing strength is an interval type-1fuzzy set, involving upper and lower firing strength, and is computed as follows [3]:

$$F^i = [\underline{f}^i, \overline{f}^i], i = 1, \cdots, M \quad (4)$$

$$\overline{f}^i = \prod_{j=1}^{n} \widehat{\overline{\mu}}_j^i, \quad \underline{f}^i = \prod_{j=1}^{n} \widehat{\underline{\mu}}_j^i \quad (5)$$

where M is the total number of rules. In (5), $\widehat{\mu}$ is the modulated membership value.

Now let us get back to the modulator functions that are to be used in the previous layer. If a feature is not useful for a rule, then we do not want that feature to influence the firing strength in (5) of the associated rule. This can be realized if for every value of that feature, both the lower membership and the upper membership values of the associated fuzzy set become 1. If a feature is bad, then we do not want it to influence the firing interval of any rule and hence none of the type-2 fuzzy sets defined on that harmful features should affect firing interval of any rule. Hence, for every value of the bad feature, both the lower membership and the upper membership values should become 1 for *all* fuzzy sets defined on that bad features. Following the idea used for Type-1 system, we use the modulator function

$$M(\lambda) = exp(-(\lambda)^2) \quad (6)$$

where λ is a modulator parameter that is a scalar variable – there will be exactly one modulator parameter for every input variable. Therefore, the modulated interval MF for the j-th feature of the i-th rule can be computed as

$$\widehat{\mu}_j^i = (\mu_j^i)^{M(\lambda_j)} = [(\underline{\mu}_j^i)^{M(\lambda_j)}, (\overline{\mu}_j^i)^{M(\lambda_j)}] \quad (7)$$

where

$$\widehat{\underline{\mu}}_j^i = (\underline{\mu}_j^i)^{M(\lambda_j)} = (\underline{\mu}_j^i)^{\exp(-(\lambda_j)^2)} \quad (8)$$

$$\widehat{\overline{\mu}}_j^i = (\overline{\mu}_j^i)^{M(\lambda_j)} = (\overline{\mu}_j^i)^{\exp(-(\lambda_j)^2)} \quad (9)$$

If modulator parameter λ_j is close to 0, then $\widehat{\mu}_j^i \approx \mu_j^i$. On the other hand, if magnitude of λ is very high, then the modulated membership value is nearly 1 ($\widehat{\mu}_j^i \approx 1$). During the learning process, we shall set appropriate values to the modulator parameters. Our goal is to make the modulator high if a feature is not useful for a rule and make it nearly zero (0) when it is useful for the rule. This is very general set up. If our intention is to select useful features, then we associate only one modulator with each feature as explained. In this case, either a feature is rejected or used by all rules. This is also done in [42] for Type-1 systems. In this paper we move one step ahead. In general, there could be a derogatory feature, which must be discarded, but there could also be a feature which is useful for some rules but not required for some other rules. This leads to the problem of selection of antecedent clause (linguistic value). Using the concept of modulator we shall learn which antecedent clause is important and which is not. If all antecedent clauses of a rule are not important, then that rule can be discarded. If some antecedent clauses are not important, then only the rule and hence the structure is simplified by removing those antecedent

clauses. If all antecedent clauses defined on a linguistic variable is not important, then that variable (feature) can be discarded.

Layer 4 (Consequent Layer): Each node in this layer represents a TSK-type consequent that is a linear combination of current input variables, where the weights of a linear combination are intervals. Each node in this layer receives inputs from layer 3 as well as the input data in layer 1. Each rule node in Layer 3 has its own corresponding consequent node in Layer 4. The output is an interval type-1 set and can be denoted as $[w_l^i, w_r^i]$, where indices l and r denote left and right limits, respectively. Therefore, the node output is expressed by

$$[w_l^i, w_r^i] = [c_0^i - s_0^i, c_0^i - s_0^i] + \sum_{j=1}^{n} [c_j^i - s_j^i, c_j^i + s_j^i] \rho_j^i \quad (10)$$

where $\rho_j^i = x_j e^{-(\lambda_j^i)^2}$, and $\rho_1^1, \rho_2^1, \cdots, \rho_N^M$ are the modulated input values. This modulation is needed because we do not want a poor feature (i.e., if it is rejected by a rule), to influence the computed consequent values. Thus,

$$w_l^i = \sum_{j=0}^{n} c_j^i \rho_j^i - \sum_{j=0}^{n} s_j^i |\rho_j^i| \quad (11)$$

and

$$w_r^i = \sum_{j=0}^{n} c_j^i \rho_j^i + \sum_{j=0}^{n} s_j^i |\rho_j^i| \quad (12)$$

where $\rho_0^i \equiv 1, i = 1, \cdots, M$.

Layer 5 (Output Processing Layer): Each node in this layer receives input from the corresponding node of layer 4 as well as that from layer 3 because to compute the output both the firing interval as well as the consequent values are needed. The output from a node in this layer is an interval type-1 set $[y_l, y_r]$, where subscripts l and r denote left and right end points, respectively. We use the K-M iterative procedure [8] to find L and R end points, and then compute the type-reduced outputs y_l and y_r. One important norm of using this procedure is that the consequent parameters must be re-ordered in ascending order, such that the original consequent parameters $w_l = (w_l^1, \cdots, w_l^M)$ and $w_r = (w_r^1, \cdots, w_r^M)$ are changed into the rule-reordered consequent parameters $v_l = (v_l^1, \cdots, v_l^M)$ and $v_r = (v_r^1, \cdots, v_r^M)$, where $v_l^1 \le v_l^2 \le \cdots \le v_l^M$ and $v_r^1 \le v_r^2 \le \cdots \le v_r^M$. The relationship between w_l, w_r, v_l and v_r can be expressed as follows

$$v_l = Q_l w_l \text{ and } v_r = Q_r w_r \quad (13)$$

where Q_l and Q_r are $M \times M$ appropriate permutation matrices to reorder the values. These two permutation matrices use elementary vectors (i.e., vectors, all of whose elements are zero except one element, which is equal to one) as columns, and these vectors are arranged (permuted) so as to move elements in w_l and w_r to new locations in ascending order in the transformed vectors v_l and v_r, respectively. Accordingly the firing strengths f^is must also be transformed to the rule-reordered firing strength g^i. Then, the outputs y_l and y_r can be computed by

$$y_l = \frac{\sum_{i=1}^{L} \overline{g}^i v_l^i + \sum_{i=L+1}^{M} \underline{g}^i v_l^i}{\sum_{i=1}^{L} \overline{g}^i + \sum_{i=L+1}^{M} \underline{g}^i} \quad (14)$$

and

$$y_r = \frac{\sum_{i=1}^{R} \underline{g}^i v_r^i + \sum_{i=R+1}^{M} \overline{g}^i v_r^i}{\sum_{i=1}^{R} \underline{g}^i + \sum_{i=R+1}^{M} \overline{g}^i}. \quad (15)$$

Layer 6 (Output layer): Each node in this layer corresponds to one output variable. Because the output of layer 5 is an interval set $[y_l, y_r]$, nodes in layer 6 compute the defuzzified value of the output linguistic variable y by computing the average of y_l and y_r as

$$y = \frac{y_l + y_r}{2} \quad (16)$$

Algorithm : Learning Algorithm IT2NFS-SIFE

IF $\vec{X}$ is the first incoming data THEN do

{Structure Learning Phase:

Generate a new rule and set initial values of relevant parameters as follows.

$[m_{j1}^1, m_{j2}^1] = [x_j - \Delta, x_j + \Delta], \sigma^1 = 0.3$ *and* $\lambda_j^1 = 1.6$;

$c_j^1 = 0.05$ *and* $s_j^1 = 0.005$; $c_0^1 = y_d$ *and* $s_0^1 = \frac{y_d}{10}$;

Compute error function using Eq. (22)

Parameter Learning Phase:

Use the rule-ordered Kalman filter algorithm (Eq. 34) to tune the consequent parameters.

And then use gradient descent algorithm (Eqs. 38-41) to update the other free parameters.

}

ELSE for each newly incoming data $\vec{X}$ do

{Structure Learning Phase:

Compute Eq. (19)

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

{M (t+1)=M (t)+1

Generate a new rule and set initial values of relevant parameters as follows:

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

$\sigma^{M(t)+1} = 0.3$ *and* $\lambda_j^{M(t)+1} = 1.6$;

$c_j^{M(t)+1} = 0.05$ *and* $s_j^{M(t)+1} = 0.005$;

$c_0^{M(t)+1} = y_d$ *and* $s_0^{M(t)+1} = \frac{y_d}{10}$;

Parameter Learning Phase:

Use the rule-ordered Kalman filter algorithm (Eq. 34) to tune the consequent parameters.

And then use gradient descent algorithm (Eqs. 38-41) to update the other free parameters.

}

ELSE

{Compute error function using Eq. (22)

Parameter Learning Phase:

Update all of relevant parameters

Use the rule-ordered Kalman filter algorithm (Eq. 34) to tune the consequent parameters.

And then use a gradient descent algorithm (Eqs. 38-41) to update the other free parameters

}

}

IV. IT2NFS-SIFE Learning

In this section, the two-phase learning scheme is discussed. Figure 3 illustrates the entire learning procedure of IT2NFS-SIFE. The first phase is on structure learning that discusses how to construct the IT2NFS-SIFE's rules. The learning starts with no rules in the IT2NFS-SIFE and all fuzzy rules are evolved by an on-line structure learning scheme. The second phase is the parameter learning phase that describes how we use a gradient

descent algorithm and the rule-ordered Kalman filter algorithm to learn the parameters of the system.

A. Structure Learning

In structure learning the system determines whether a new rule should be extracted and added from the training data or not. This idea is extended from our previous study on type-1 fuzzy rule generation [51]. The first incoming data point $\vec{x}$ is used to generate the first fuzzy rule with mean and width of the fuzzy membership functions as

$$[m_{j1}^1, m_{j2}^1] = [x_j - \Delta x, x_j + \Delta x] \text{ and } \sigma = \sigma_{\text{fixed}} \quad j = 1, \cdots, n, \tag{17}$$

where Δx is a constant to determine the initial type-2 fuzzy set interval range and σ_{fixed} is a predefined value (we use σ_{fixed} = 0.3 in this paper). The Δx indicates the width of uncertain mean region. If Δx is made too small, then the type-2 fuzzy set approximately becomes a type-1 fuzzy set. On the contrary, if Δx is too large, then a type-2 fuzzy set will cover most of the input domain. Subsequently, at time t, when a new input comes, we compute the firing strengths (firing intervals) of that input for all M(t) rules and then compute the mean value of firing strength $f_c^i(t)$ for each rule i as:

$$f_c^i(t) = \frac{\underline{f^i(t)} + \overline{f^i(t)}}{2} \tag{18}$$

f_c helps us to determine whether a new rule should be generated according to a predefined threshold f_{th}.

Thus, for subsequent incoming data, we find

$$I = \arg \max_{1 \le i \le M(t)} f_c^i(t), \tag{19}$$

where M(t) is the number of existing rules at time t. If $f_c^I(t) \le f_{th}$, then a new fuzzy rule is generated and M(t+1) = M(t)+1. The idea behind this is that if the current data point does not match well the existing rules, then a new fuzzy rule is evolved. The uncertain means of type-2 fuzzy sets associated with this rule are defined exactly in the same manner as done for the first rule; i.e., the uncertain means of the corresponding type-2 fuzzy sets are defined as

$$[m_{j1}^{M(t)+1}, m_{j2}^{M(t)+1}] = [x_j(t) - \Delta x, x_j(t) + \Delta x], j = 1, \cdots, n \tag{20}$$

But the width of the new rule is defined as follows:

$$\sigma_j^{M(t)+1} = \beta \left| x_j - \left(\frac{m_{j1}^I + m_{j2}^I}{2} \right) \right| \tag{21}$$

where $\beta > 0$ determines the overlap degree between two fuzzy sets. Equation (21) indicates that the initial width is equal to the Euclidean distance between current input data $\vec{x}$ and the center of the best matching rule for this data point times an overlapping parameter. Here, β is set to 0.5 in this paper and it indicates that the width of new type-2 fuzzy set is half of the Euclidean distance from the best matching center, and a suitable overlap between adjacent rules is realized. Similar protocols were also followed in [16], [18], [27].

In addition to assigning the initial antecedent parameters, the initial consequent parameters should also be assigned when a new rule is generated. The initial consequent parameters c_0^i, $i = 1, \cdots, M$, are set to the desired output y_d corresponding to the current input $\vec{x}$. Other initial parameters c_j^i, $j = 1, \cdots, N$, are set to small values, e.g. 0.05. A parameter S is used to define the output interval range. If the interval range is too small, the output is similar to singleton. On the other hand, if the interval range is too large, then it would cover the whole output domain. In this study we use parameters $s_j^i = 0.005$, $j = 1, \cdots, n$ for all variables.

B. Parameter Learning

The parameter learning is also done during the learning of the structure. All free parameters of the IT2NFS-SIFE are adjusted with each incoming training data regardless of whether a rule is newly generated or already existing. The antecedent parameters, involving m_1, m_2, and λ, in the IT2NFS-SIFE are adjusted with each incoming input by gradient descent algorithm that is suitable for supervised method. The consequent part parameters are tuned through a rule-ordered Kalman filter algorithm to improve the network accuracy. Here we consider the single output case just for clarity. The training algorithm minimizes the error function

$$E = \frac{1}{2}[y(t+1) - y_d(t+1)]^2, \tag{22}$$

where $y(t+1)$ represents the actual network output and $y_d(t+1)$ represents the desired output. Note that (22) represents the instantaneous square error and with each incoming data point, only one step of update of all parameters is done using gradient descent (the learning rules will be derived later). As reported in [17], a rule-ordered Kalman filter algorithm is applied for the learning of the consequent parameters. Next, we describe the rule-ordered Kalman filter algorithm explicitly. Based on K-M iterative procedure for computing y_l and y_r in (14) and (15), the consequent values w_l and w_r should be re-arranged in ascending order. As the consequent values w_l and w_r change, their orders and corresponding rule orders should also change accordingly. Equation (13) indicates the relationship between the original and arranged values. As in [1], Eqns. (14) and (15) can be re-expressed as the following rule-ordered form.

$$y_l = \frac{\overline{f}^T Q_l^T E_1^T E_1 Q_l w_l + \underline{f}^T Q_l^T E_2^T E_2 Q_l w_l}{\sum_{i=1}^{L} (Q_l \overline{f})_i + \sum_{i=L+1}^{M} (Q_l \underline{f})_i} \tag{23}$$

and

$$y_r = \frac{\underline{f}^T Q_r^T E_3^T E_3 Q_r w_r + \overline{f}^T Q E_4^T E_4 Q_r w_r}{\sum_{i=1}^{R} (Q_r \underline{f})_i + \sum_{i=R+1}^{M} (Q_r \overline{f})_i} \tag{24}$$

where

$$E_1 = (e_1, e_2, \cdots, e_L, 0, \cdots, 0) \in \Re^{L \times M}, \tag{25}$$

$$E_2 = (0, \cdots, 0, \varepsilon_1, \varepsilon_2, \cdots, \varepsilon_{M-L}) \in \Re^{(M-L) \times M}, \tag{26}$$

$$E_3 = (e_1, e_2, \cdots, e_R, 0, \cdots, 0) \in \Re^{R \times M}, \tag{27}$$

$$E_2 = (0, \cdots, 0, \varepsilon_1, \varepsilon_2, \cdots, \varepsilon_{M-R}) \in \Re^{(M-R) \times M}, \tag{28}$$

and e_i and ε_i are unit vectors. The rule-ordered Kalman filter algorithm tunes the consequent parameters in the IT2NFS-SIFE. Therefore, Eqs. (23) and (24) can be re-expressed as $y_l = \phi_l^T w_l$,

where $\phi_l = \frac{\overline{f}^T Q_r^T E_1^T E_1 Q_l + \underline{f}^T Q_l^T E_2^T E_2 Q_l}{\sum_{i=1}^{L}(Q_r \overline{f})_i + \sum_{i=L+1}^{M}(Q_l \underline{f})_i} \in \Re^{M\times 1}$ (29)

$y_r = \phi_r^T w_r$,

where $\phi_r = \frac{\underline{f}^T Q_r^T E_3^T E_3 Q_r + \overline{f}^T Q_r^T E_4^T E_4 Q_r}{\sum_{i=1}^{R}(Q_r \underline{f})_i + \sum_{i=R+1}^{M}(Q_r \overline{f})_i} \in \Re^{M\times 1}$ (30)

Thus, the output y in (16) can be re-expressed as

$$y = \frac{y_l + y_r}{2} = \frac{1}{2}(\phi_l^T w_l + \phi_r^T w_r) = [\overline{\phi}_l^T \ \overline{\phi}_r^T]\begin{bmatrix} w_l \\ w_r \end{bmatrix}$$

$$= [\overline{\phi}_{l1}\cdots\overline{\phi}_{lM} \ \overline{\phi}_{r1}\cdots\overline{\phi}_{rM}]\begin{bmatrix} \sum_{j=0}^{n} c_j^1\rho_j^1 - \sum_{j=0}^{n} s_j^1\left|\rho_j^1\right| \\ \vdots \\ \sum_{j=0}^{n} c_j^M\rho_j^M - \sum_{j=0}^{n} s_j^M\left|\rho_j^M\right| \\ \sum_{j=0}^{n} c_j^1\rho_j^1 + \sum_{j=0}^{n} s_j^1\left|\rho_j^1\right| \\ \vdots \\ \sum_{j=0}^{n} c_j^M\rho_j^M + \sum_{j=0}^{n} s_j^M\left|\rho_j^M\right| \end{bmatrix} \quad (31)$$

where $\overline{\phi}_l = 0.5\phi_l$ and $\overline{\phi}_r = 0.5\phi_r$. During the on-line structure learning, the dimension of w_l and w_r increases with time, and the positions of c_j^i and s_j^i change accordingly within the same vector. For keeping the position of c_j^i and s_j^i unaltered in the vector, the rule-ordered Kalman filtering algorithm rearranges elements in rule order in (31). Let $w_{TSK} \in \Re^{2M(n+1)}$ denote all of consequent parameters. That is

$$w_{TSK} = [c_0^1\cdots c_n^1 \ s_0^1\cdots s_n^1\cdots c_0^M\cdots c_n^M \ s_0^M\cdots s_n^M]^T \quad (32)$$

where the consequent parameters are placed according to the rule order. Equation (31) can then be expressed as

$$\begin{aligned} y &= \overline{\phi}_{TSK}^T w_{TSK} \\ &= [\overline{\phi}_{c1}\rho_0^1\cdots\overline{\phi}_{c1}\rho_n^1 \ -\overline{\phi}_{s1}\left|\rho_0^1\right|\cdots-\overline{\phi}_{s1}\left|\rho_n^1\right| \ \cdots \\ &\quad \overline{\phi}_{cM}\rho_0^M\cdots\overline{\phi}_{cM}\rho_n^M \ -\overline{\phi}_{sM}\left|\rho_0^M\right|\cdots-\overline{\phi}_{sM}\left|\rho_n^M\right|]w_{TSK} \end{aligned} \quad (33)$$

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

$$w_{TSK}(t+1) = w_{TSK}(t) + S(t+1)\overline{\phi}_{TSK}(t+1)(y_d(t+1) - \overline{\phi}_{TSK}^T(t+1)w_{TSK}(t))$$

$$S(t+1) = \frac{1}{\kappa}[S(t) - \frac{S(t)\overline{\phi}_{TSK}(t+1)\overline{\phi}_{TSK}^T(t+1)S(t)}{\kappa + \overline{\phi}_{TSK}^T(t+1)S(t)\overline{\phi}_{TSK}(t+1)}] \quad (34)$$

where κ is a forgetting factor ($0 < \kappa \le 1$). In this study, we have used $\kappa = 1$. The dimensions of vectors w_{TSK} and $\overline{\phi}_{TSK}$, and matrix S increases when a new rule is evolved. When a new rule is generated at time $t+1$, the dimension of $\overline{\phi}_{TSK}$, w_{TSK}, and $S(t)$ can be expanded and expressed by the following formulas.

$$\overline{\phi}_{TSK}^T(t+1) = [\overline{\phi}_{TSK}^T(t) \ \overline{\phi}_{c(M+1)}x_0\cdots\overline{\phi}_{c(M+1)}x_n \ -\overline{\phi}_{s(M+1)}\left|x_0\right|\cdots\overline{\phi}_{s(M+1)}\left|x_n\right|] \in \Re^{2(M+1)(n+1)} \quad (35)$$

$$w_{TSK}(t) = [w_{TSK}(t) \ c_0^{M+1}\cdots c_n^{M+1} \ s_0^{M+1}\cdots s_n^{M+1}] \in \Re^{2(M+1)(n+1)} \quad (36)$$

and

$$S(t) = blockdiag[S(t) \ CI] \in R^{2(M+1)(n+1)\times 2(M+1)(n+1)} \quad (37)$$

where C is a large positive constant and I is an identity matrix. Note that the dimension of initial S is a 1x1. For convergence analysis of the Kalman filter with varying input dimensions, the proposed network uses resetting of the matrix S to CI after ten iterations of learning. This resetting operation keeps S bounded and helps avoid a divergence problem.

In previous studies [1], [8], [17], a gradient descent algorithm is used for the parameter learning, and let m_{j1}^i, m_{j2}^i, σ_j^i, and λ_j^i represent the antecedent and modulator parameters, then the learning rules are:

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

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

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

$$\lambda_j^i(t+1) = \lambda_j^i(t) - \eta'\frac{\partial E}{\partial \lambda_j^i(t)} \quad (41)$$

where η and η' are learning coefficients for the antecedent parameters and the modulators respectively. Now we can compute the learning rules as follows:

$$\begin{aligned} \frac{\partial E}{\partial \lambda_j^i} &= \frac{\partial E}{\partial y}\left(\frac{\partial y}{\partial y_l}\frac{\partial y_l}{\partial \lambda_j^i} + \frac{\partial y}{\partial y_r}\frac{\partial y_r}{\partial \lambda_j^i}\right) \\ &= \frac{1}{2}(y - y_d)\left[(\frac{\partial y_l}{\partial \overline{f}^i} + \frac{\partial y_r}{\partial \overline{f}^i})\frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \lambda_j^i} + (\frac{\partial y_l}{\partial \underline{f}^i} + \frac{\partial y_r}{\partial \underline{f}^i})\frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \lambda_j^i}\right] \end{aligned} \quad (42)$$

and where

$$\frac{\partial y_l}{\partial \overline{f}^i} = \frac{a_l^i - y_l c_l^i}{\overline{f}^T c_l + \underline{f}^T d_l}, \ \frac{\partial y_r}{\partial \overline{f}^i} = \frac{b_r^i - y_r d_r^i}{\underline{f}^T c_r + \overline{f}^T d_r} \quad (43)$$

$$\frac{\partial y_l}{\partial \underline{f}^i} = \frac{b_l^i - y_l d_l^i}{\overline{f}^T c_l + \underline{f}^T d_l}, \ \frac{\partial y_r}{\partial \underline{f}^i} = \frac{a_r^i - y_r c_l^i}{\underline{f}^T c_r + \overline{f}^T d_r} \quad (44)$$

$$\frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \lambda_j^i} = -2\times\overline{f}^i\times\lambda_j^i\times e^{-(\lambda_j^i)^2}\times\ln\left|\overline{\mu}_j^i\right| \quad (45)$$

$$\frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \lambda_j^i} = -2\times\underline{f}^i\times\lambda_j^i\times e^{-(\lambda_j^i)^2}\times\ln\left|\underline{\mu}_j^i\right| \quad (46)$$

where

$$a_l = Q_l^T E_1^T E_1 Q_l w_l \in \Re^{M\times 1}, \ b_l = Q_l^T E_2^T E_2 Q_l w_l \in \Re^{M\times 1} \quad (47)$$

$$c_l = Q_l^T p_l \in \Re^{M\times 1}, \ d_l = Q_l^T g_l \in \Re^{M\times 1} \quad (48)$$

$$a_r = Q_r^T E_3^T E_3 Q_r w_r \in \Re^{M\times 1},\ b_r = Q_r^T E_4^T E_4 Q_r w_r \quad (49)$$

$$c_r = Q_r^T p_r \in \Re^{M\times 1},\ d_r = Q_r^T g_r \in \Re^{M\times 1} \quad (50)$$

Similarly, if w_j^i represents m_{j1}^i, m_{j2}^i, and σ_j^i, the derivation of premise parts can be expressed as

$$\frac{\partial E}{\partial w_j^i} = \frac{\partial E}{\partial y}\left(\frac{\partial y}{\partial y_l}\frac{\partial y_l}{\partial w_j^i} + \frac{\partial y}{\partial y_r}\frac{\partial y_r}{\partial w_j^i}\right)$$
$$= \frac{1}{2}(y-y_d)\left[\left(\frac{\partial y_l}{\partial \overline{f}^i} + \frac{\partial y_r}{\partial \overline{f}^i}\right)\frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial w_j^i} + \left(\frac{\partial y_l}{\partial \underline{f}^i}\frac{\partial y_r}{\partial \underline{f}^i}\right)\frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial w_j^i}\right] \quad (51)$$

If $w_j^i = m_{j1}^i$, then we have

$$\frac{\partial \overline{f}^i}{\partial w_j^i} = \frac{\partial \overline{f}^i}{\partial m_{j1}^i} = \frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j1}^i} = \begin{cases} \overline{f}^i \times e^{-(\lambda_j^i)^2} \times \overline{\mu}_j^i \times \dfrac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, & x_j \le m_{j1}^i \\ 0, & \text{otherwise} \end{cases} \quad (52)$$

$$\frac{\partial \underline{f}^i}{\partial w_j^i} = \frac{\partial \underline{f}^i}{\partial m_{j1}^i} = \frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j1}^i} = \begin{cases} \underline{f}^i \times e^{-(\lambda_j^i)^2} \times \underline{\mu}_j^i \times \dfrac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ 0, & \text{otherwise} \end{cases} \quad (53)$$

Consequently, if $w_j^i = m_{j2}^i$, then we have

$$\frac{\partial \overline{f}^i}{\partial w_j^i} = \frac{\partial \overline{f}^i}{\partial m_{j2}^i} = \frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j2}^i} = \begin{cases} \overline{f}^i \times e^{-(\lambda_j^i)^2} \times \overline{\mu}_j^i \times \dfrac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, & x_j > m_{j1}^i \\ 0, & \text{otherwise} \end{cases} \quad (54)$$

$$\frac{\partial \underline{f}^i}{\partial w_j^i} = \frac{\partial \underline{f}^i}{\partial m_{j2}^i} = \frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j2}^i} = \begin{cases} \underline{f}^i \times e^{-(\lambda_j^i)^2} \times \underline{\mu}_j^i \times \dfrac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ 0, & \text{otherwise} \end{cases} \quad (55)$$

Finally, if $w_j^i = \sigma_j^i$, then we have

$$\frac{\partial \overline{f}^i}{\partial w_j^i} = \frac{\partial \overline{f}^i}{\partial \sigma_j^i} = \frac{\partial \overline{f}^i}{\partial \hat{\overline{\mu}}_j^i}\frac{\partial \hat{\overline{\mu}}_j^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial \sigma_j^i} = \begin{cases} \overline{f}^i \times e^{-(\lambda_j^i)^2} \times \overline{\mu}_j^i \times \dfrac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j < m_{j1}^i \\ \overline{f}^i \times e^{-(\lambda_j^i)^2} \times \overline{\mu}_j^i \times \dfrac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j > m_{j2}^i \\ 0, & \text{otherwise} \end{cases} \quad (56)$$

$$\frac{\partial \underline{f}^i}{\partial w_j^i} = \frac{\partial \underline{f}^i}{\partial \sigma_j^i} = \frac{\partial \underline{f}^i}{\partial \hat{\underline{\mu}}_j^i}\frac{\partial \hat{\underline{\mu}}_j^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \sigma_j^i} = \begin{cases} \underline{f}^i \times e^{-(\lambda_j^i)^2} \times \underline{\mu}_j^i \times \dfrac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ \underline{f}^i \times e^{-(\lambda_j^i)^2} \times \underline{\mu}_j^i \times \dfrac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \end{cases} \quad (57)$$

For each piece of incoming data only one epoch of the gradient descent algorithm and the rule-ordered Kalman filter algorithm is performed irrespective of whether it is an old sample from repeated offline learning or a new sample from online learning. To begin the learning process, we need to initialize the modulator parameters, λs. For an effective learning, we initialize the λ values in such a manner that at the beginning all features as well as all antecedent clauuses are unimportant, i.e., all modulated membership values are close to 1. So we set the initial λ values to 1.6. Then with iteration, the modulators for useful antecedent clauses are adapted (reduced) faster as those can reduce the error, while for bad features/not-useful antecedent clauses the values of λ will increase and the modulated membership will approach 1. The entire learning process is summarized as an algorithm named "Learning Algorithm IT2NFS-SIFE". This Algorithm specifies how a rule and its membership functions are initialized.

V. Simulations

In this section, we describe four examples: System identification (Example 1), Hang data (Example 2), Chem data (Example 3) and Auto MPG6 (Example 4), to assess the performance of the IT2NFS-SIFE. Table I shows the summary of the dataset used. Our simulation studies demonstrate that the IT2NFS-SIFE is able to effectively prune redundant features and simplify structure simultaneously with online learning of the system.

A. Example 1 (System Identification)

This example uses the IT2NFS-SIFE to demonstrate that the use of modulator function is able to remove redundant inputs and simplify the network architecture. The plant has been presented in [17], [51], and is described by the difference equation

TABLE I
SUMMARY OF THE DATA SETS USED

Data	No. of Samples	No. of Features	No. of derogatory Features
SINC	200	4	2
HANG	50	4	2
CHEM	70	5	2
Auto-MPG	392	7	2

$$y_d(t+1) = \frac{y_d(t)}{1+y_d^2(t)} + u^3(t) \quad (58)$$

where the training signal is generated with $u(t) = sin(2\pi t/100)$, $t = 1, \cdots, 200$. The inputs in the IT2NFS-SIFE are $u(t)$ and $y_d(t)$ and the desired output is $y_d(t+1)$. Here for observing the merit of IT2NFS-SIFE, we artificially generate two irrelevant input variables. These derogatory inputs are scaled in the range [0, 1]. As a performance criterion, we use the root mean square error (RMSE)

$$RMSE = \sqrt{\frac{1}{200}\sum_{t=1}^{200}[y(t+1) - y_d(t+1)]} \quad (59)$$

where y(t + 1) represents IT2NFS-SIFE output. The structure threshold and learning rate are set to 0.02 and 0.07, respectively. The learning rate for modulator parameters is set to 2×10^{-4}. Training is performed for 200 epochs each with 200 steps and three fuzzy rules are generated. After training, the RMSE obtained is 0.018. Figure 4 shows that the convergence curve of training process is quite fast. Fig. 5 shows a very good match between the actual output and the network output. For this example we also compare the performance of IT2NFS-SIFE with two well-known models as shown in Table II. Table II exhibits the network performance, including the number of rules, test RMSE and compares them with its competitors presented in [17], [51]. Table II reveals that IT2NFS-SIFE

outperforms the other competitors. To compare with a feed-forward type-1 FNN, we use a self-organizing neural fuzzy inference network (SONFIN) [51], which is a powerful network with both structure and parameter learning abilities. The consequent part of the SONFIN is of Mamdani type. All free parameters in SONFIN are learned using a gradient descent algorithm. We also compare with another a feed forward type-2 FNN, which is the self-evolving interval type-2 fuzzy neural

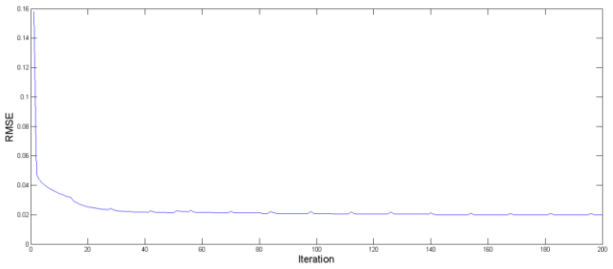

Fig. 4. RMSE values obtained during training process (Example 1)

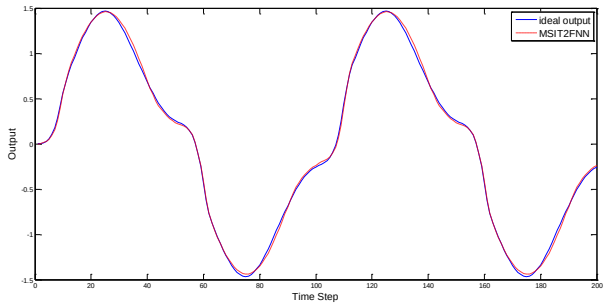

Fig. 5. Comparison of the outputs of IT2NFS-SIFE with the actual values using all four input variables (Example 1)

network (SEIT2FNN) [17]. This network too possesses simultaneous structure and parameter learning abilities. The consequent part in the IT2NFS-SIFE is of Takagi–Sugeno–Kang (TSK) type. Parameter learning philosophy of SEIT2FNN is the same as that of IT2NFS-SIFE. These two well-known models are applied to investigate whether two invalid inputs will significantly affect the final output. The test root mean square error (RMSE) of SONFIN is obtained as 0.0198 when inputs do not involve two invalid variables. The test RMSE of the SEIT2FNN is obtained as 0.0166 with three fuzzy rules. From these results, we see that SEIT2FNN outperforms SONFIN. Table II shows that the RMSE of SONFIN and SEIT2FNN respectively increases to 0.088 and 0.071. Thus, the two irrelevant inputs significantly affect the identified systems and their outputs and add to computational complexity. But our proposed network could effectively reduce the impact of the two derogatory inputs without degrading the network performance.

Here, we can see an interesting phenomenon in Table III - the modulator parameter for all antecedent clauses defined on the irrelevant inputs (x_3 and x_4) have increased from their original value of 1.6 and they have become bigger than the modulator values for all antecedent clauses defined on the original inputs (x_1 and x_2). This suggests that we can discard both x_3 and x_4 and simplify the network structure. Since there is no rule for which all modulator parameters are high, we cannot delete any rule.

B. Example 2 (Hang Data)

Now we consider the Hang data [32], [42] that is a synthetic data set to demonstrate the distinguished characteristic of IT2NFS-SIFE. This data set is generated by the following equation

$$y = (1 + x_1^{-1} + x_2^{-1.5})^2, \qquad 1 \le x_1, x_2 \le 5. \tag{60}$$

where x_1 and x_2 are input variables for the IT2NFS-SIFE input layer and y represents a single output. To obtain the training data, Equation (60) has generated 50 I-O data points randomly from $1 \le x_1, x_2 \le 5$. To show the capability of our system, we have augmented this data set by adding two useless variables x_3 and x_4 with random values in [1, 5]. Fig. 6 depicts a pictorial representation of the Hang data. The structure threshold and learning rate are set as 0.1 and 0.07, respectively. The learning rate for modulator function is set to 7×10^{-4}.

After 200 epochs of training, three rules are generated. Like

TABLE II
PEFFORMANCE COMPARISON OF IT2NFS-SIFE AND OTHER MODLES IN EXAMPLE 1

Models	SONFIN [51]	SEIT2FNN [17]	IT2NFS-SIFE
Iterations	200	200	200
No. of Rules	6	3	3
Test RMSE	0.088	0.071	0.021

TABLE III
MODULATOR VALUES ASSOCIATED WITH INPUTS AND RULES FOR EXAMPLE 1 FOR A TYPICAL RUN

λ	Rule 1	Rule 2	Rule 3
x_1	1.064	0.723	0.976
x_2	1.104	1.042	0.981
x_3	1.734	1.643	1.672
x_4	1.658	1.788	1.776

TABLE IV
MODULATOR VALUES ASSOCIATED WITH INPUTS AND RULES FOR EXAMPLE 1 WHEN 7 RULES ARE USED

λ	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
x_1	1.066	1.024	1.045	1.037	1.063	0.873	0.934
x_2	1.006	0.919	1.155	1.057	1.097	1.005	0.933
x_3	1.625	1.615	1.612	1.580	1.627	1.663	1.591
x_4	1.638	1.639	1.622	1.564	1.678	1.603	1.599
Train RMSE	0.0057						
Test RMSE	0.0062						

the previous example, we compare the performance of IT2NFS-SIFE with that of SONFIN and SEIT2FNN. As used by IT2NFS-SIFE these two models, SONFIN and SEIT2FNN, employ the identical data set obtained from [32]. The RMSE on the test data for the SONFIN and SEIT2FNN are respectively 0.226 and 0.207, when they use only the two useful (original) features x_1 and x_2. When the irrelevant features are used by these two systems, the test RMSE increases to 0.385 and 0.313 as shown in Table V. But using all four features, the test RMSE of our system is only 0.228. We note that the results in Table V are obtained using a five-fold cross-validation framework as we have fixed data set without any training-test partition. Thus from Table V we find that the performance of SONFIN and SEIT2FNN is severely affected by the presences of derogatory features, but our system is not. The modulator values in Table VI indicate that x_3, x_4 are bad features and should be discarded

and $Rule_3$ is also not important as it does not affect the system output and hence can be discarded without any loss. Note that, the earlier attempts that use modulators for Type-1 fuzzy systems do not have the capability of removing a rule.

C. Example 3 (Chem Data)

This example uses the IT2NFS-SIFE to address a real-world problem relating to operator's control of a chemical plant for producing a polymer by polymerization of some monomers [32]. There are five inputs and one output. The input variables are: monomer concentration (u_1), change of monomer concentration (u_2), monomer flow rate (u_3), and two local temperatures inside the plant (u_4 and u_5). The output y represents the set point for monomer flow rate. This data set contains 70 samples. In [32], the authors indicate that u_4 and u_5 do not influence the system output. Various other authors [32], [42] also confirmed the same observation. Hence for this data set we do not add any extra feature and use the five features to analyze the influence of irrelevant features on our network. The

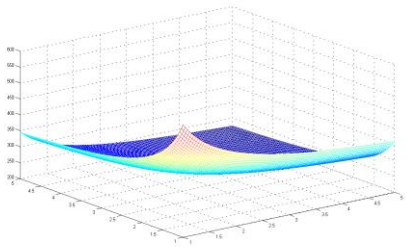

Fig. 6 A pictorial representation of HANG (Example 2)

TABLE V
PEFFORMANCE COMPARISON OF IT2NFS-SIFE AND OTHER MODLES FOR HANG DATA IN EXAMPLE 2

Models	SONFIN [51]	SEIT2FNN [17]	IT2NFS-SIFE
Iterations	100	100	100
No. of Rules	5	3.2	3.2
Test RMSE	0.385	0.313	0.228

TABLE VI
COEIFFIEIENT OF MODULATOR FUNCTION ASSOCIATE WITH INPUTS AND RULES for EXAMPLE 2

λ	Rule 1	Rule 2	Rule 3
x_1	1.333	-0.407	-4.000
x_2	-0.027	0.825	-4.000
x_3	2.417	2.812	4.000
x_4	4.000	4.000	4.000

structure threshold and learning rate for modulators for this data set are set to 0.4 and 3×10^{-4}, respectively. There are 4 fuzzy rules generated after 100 epochs of training. Fig. 7 depicts the outputs of the Chem data and those of the IT2NFS-SIFE. The RMSE on this data set for SONFIN and SEIT2FNN, without involving two local temperatures, are respectively 148.64 and 84.37. As depicted in Table VII these RMSEs enhances to 193.74 and 126.25 for SONFIN and SEIT2FNN, respectively when all five features are used. Table VII depicts that when all five features are used, then the RMSEs for SONFIN and SEIT2FNN are more than 5.3 and 3.4 times respectively as that of IT2NFS-SIFE. The terminal modulator values in Table VIII also suggests that the two temperatures, u_4 and u_5, are not

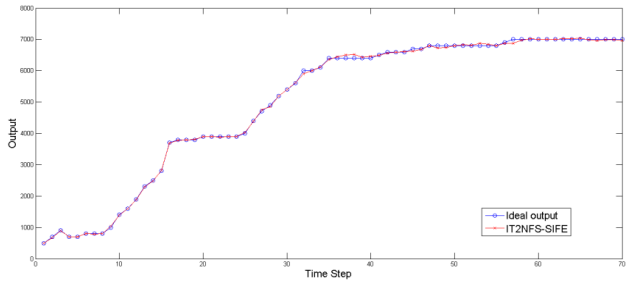

Fig. 7. Plot of the estimated results and the actual observed values for the Chem data (Example 3)

TABLE VII
PERFORMANCE COMPARISON OF IT2NFS-SIFE AND OTHER MODLES IN EXAMPLE 3

Models	SONFIN [51]	SEIT2FNN [17]	IT2NFS-SIFE
Iteration	100	100	100
No. of Rules	6	4	4
Test RMSE	193.74	126.25	36.76

TABLE VIII
COEIFFIEIENT OF MODULATOR FUNCTION ASSOCIATE WITH INPUTS AND FUZZY RULES FOR EXAMPLE 3

λ	Rule 1	Rule 2	Rule 3	Rule 4
u_1	0.600	0.543	1.156	0.601
u_2	0.533	-0.296	1.120	0.602
u_3	-0.062	0.443	0.385	0.602
u_4	-1.902	1.988	2.905	2.202
u_5	1.732	1.684	3.306	2.201

important as the absolute values of their modulators for all rules are high.

D. Example 4 (Auto-MPG6)

This is a real dataset that involves the prediction of automobile city-cycle fuel consumption in miles per gallon (MPG). The data set contains 392 samples and can be downloaded from University of California-Irvine (UCI) (http://archive.ics.uci.edu/ml/). As done in [59], [60], the data set divided two subsets with 320 samples for training and remaining 72 samples for test. There are seven input variables, involving cylinders (x_1), displacement (x_2), horsepower (x_3), weight (x_4), acceleration (x_5), model year (x_6) and origin (x_7), and one output variable, mpg (y). It is evident from the description of the input features that x_6 and x_7 are not relevant features and that is why other studies such as in [61], these two features are ignored. But we shall use all seven features to investigate whether our system can eliminate these to irrelevant features or not. The structure threshold and learning rate for modulators are used as 0.005 and 6×10^{-5}, respectively. In all other examples, the learning rates for the premise and consequent parameters are taken as 0.07 but for this example, we are required to use a lower value of the learning coefficient for a smooth convergence of the training. After 100 epochs of training, there are three fuzzy rules generated. When the system uses the five features, $(x_1,\cdots,x_5)$, the RMSEs obtained for SONFIN and SEIT2FNN are 2.98 and 2.81, respectively. In [61], the authors also use five significant features $(x_1,\cdots,x_5)$ to predict automobile city-cycle fuel consumption.

As can be seen in Table IX, both for SONFIN and SEIT2FNN use of the two irrelevant features leads to a

degradation in performance. As revealed by Table IX the test RMSE of IT2NFS-SIFE can obtain lower than its competitors. Table X shows that the input variables x_6 and x_7 are not important as their modulator values are high for all rules. Table X shows that the modulation parameters of x_6 and x_7 are bigger than those of other input variables, and therefore, input variables x_6 and x_7 can be discarded to enhance performance further.

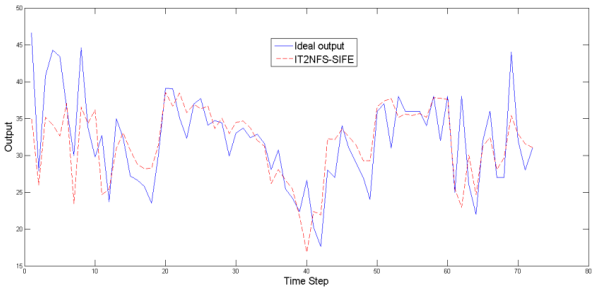

Fig. 8. Plot of the estimated results and the actual observed values for the auto-MPG data (Example 4)

TABLE IX
PERFORMANCE COMPARISON OF IT2NFS-SIFE AND OTHER MODLES IN EXAMPLE 4

Models	RBFNN-SFC [59]	SONFIN [51]	SEIT2FNN [17]	IT2NFS-SIFE
Iteration	-	100	100	100
Number of Rules	6	5	3	3
Test RMSE	6.77	4.66	4.45	4.23

TABLE X
MODULATOR VALUES ASSOCIATED WITH INPUTS AND RULES FOR EXAMPLE 4

λ	Rule 1	Rule 2	Rule 3
x_1	0.640	0.633	0.632
x_2	0.639	0.631	0.736
x_3	1.137	0.729	0.628
x_4	0.842	0.781	1.038
x_5	0.740	0.932	0.887
x_6	1.621	1.562	1.604
x_7	1.648	1.704	1.699

VI. Conclusion

In this paper, we propose new interval type-2 neural fuzzy system called IT2NFS-SIFE that can reduce the effect of bad or derogatory features (in fact can remove bad features), it can also simply network structure by removing antecedent clauses that are not important for a rule. Unlike most self-organizing fuzzy neural networks, it has the flexibility of learning different rules involving different sets of variables, if that makes the system identification easier. To our knowledge, this is the first Type-2 fuzzy neural system that can selects features and adapts its structure and learn parameters simultaneously. It uses the concept of membership modulators that eliminate the impact of a type-2 fuzzy membership function on the output if that membership function (hence the associated antecedent clause) is not useful. The self-evolving ability in our network enables it to efficiently identify the required structure of the network and does not need to set any initial IT2NFS-SIFE structure in advance. For the parameter learning in the IT2NFS-SIFE, the antecedent part and modulation parameters are trained using the error back-propagation learning, and the rule-ordered Kalman filter algorithm helps improve network accuracy by tuning the consequent part parameters. To demonstrate the effectiveness of IT2NFS-SIFE we have tested it on four commonly used data sets and compared its performance with that of two competitive algorithms. For all four case, not only system was able to yield better performance in terms of RMSE, but also could simply the system drastically. IT2NFS-SIFE was able identify the irrelevant features as well as irrelevant rules and irrelevant clauses.

References

[1] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 12, 2004.

[2] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. Fuzzy Syst.*, vol. 8, 2000.

[3] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 535–550, 2000.

[4] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 643–658, 1999.

[5] R. John and S. Coupland, "Type-2 Fuzzy Logic: A Historical View," *IEEE Comput. Intell. Mag.*, vol. 2, 2007.

[6] J. M. Mendel, "Type-2 Fuzzy Sets and Systems: An Overview," *IEEE Comput. Intell. Mag.*, vol. 2, 2007.

[7] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, 2002.

[8] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, vol. 2. 2001, p. 555.

[9] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," *Inf. Sci.*, vol. 177, pp. 84–110, 2007.

[10] J. M. Mendel and R. I. B. John, "Type-2 Fuzzy Sets Made Simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 117–127, 2002.

[11] 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, 2010.

[12] C.-H. Lee, Y.-C. Lin, and W.-Y. Lai, "Systems identification using type-2 fuzzy neural network (type-2 FNN) systems," *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, vol. 3. pp. 1264–1269 vol. 3, 2003.

[13] J. Zeng and Z.-Q. Liu, "Type-2 fuzzy hidden Markov models and their application to speech recognition," *IEEE Trans. Fuzzy Syst.*, vol. 14, 2006.

[14] C. Lee, J. Hong, Y. Lin, and W. Lai, "Type-2 fuzzy neural network systems and learning," vol. 1, no. 4, pp. 79–90, 2003.

[15] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, 2004.

[16] 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, 2009.

[17] C. -F. Juang and Y. -W. Tsao, "A Self-Evolving Interval Type-2 Fuzzy Neural Network With Online Structure," *IEEE Trans. Fuzzy Syst.*, vol. 16, pp. 1411–1424, 2008.

[18] 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, pp. 447–459, 2014.

[19] F.-J. Lin and P.-H. Chou, "Adaptive Control of Two-Axis Motion Control System Using Interval Type-2 Fuzzy Neural Network," *IEEE Trans. Ind. Electron.*, vol. 56, 2009.

[20] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems to forecasting of time-series," *Inf. Sci. (Ny).*, vol. 120, pp. 89–111, 1999.

[21] H. Hagras, "Comments on “Dynamical Optimal Training for Interval Type-2 Fuzzy Neural Network (T2FNN)”," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 36, 2006.

[22] O. Castillo and P. Melin, *Comparison of Hybrid Intelligent Systems, Neural Networks and Interval Type-2 Fuzzy Logic for Time Series Prediction*. 2007, pp. 3086–3091.

[23] C.-H. Wang, C.-S. Cheng, and T.-T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *SMC '03 Conf. Proceedings. 2003 IEEE Int. Conf. Syst. Man Cybern. Conf. Theme - Syst. Secur. Assur. (Cat. No.03CH37483)*, vol. 4, 2003.

[24] Y. Özbay, R. Ceylan, and B. Karlik, "Integration of type-2 fuzzy clustering and wavelet transform in a neural network based ECG classifier," *Expert Syst. Appl.*, vol. 38, pp. 1004–1010, 2011.

[25] J. R. Castro, O. Castillo, P. Melin, A. Rodriguez-Diaz, and L. G. Martinez, "Intelligent Control Using an Interval Type-2 Fuzzy Neural Network with a Hybrid Learning Algorithm," *2008 Ieee Int. Conf. Fuzzy Syst. Vols 1-5*, pp. 893–900\n2438, 2008.

[26] 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. (Ny).*, vol. 179, pp. 2175–2193, 2009.

[27] Y. Lin, J. Chang, N. R. Pal, and C. Lin, "A Mutually Recurrent Interval Type-2 Neural Fuzzy Structure and Parameters," vol. 21, no. 3, pp. 492–509, 2013.

[28] G. M. Méndez and M. De Los Angeles Hernández, "Hybrid learning mechanism for interval A2-C1 type-2 non-singleton type-2 Takagi-Sugeno-Kang fuzzy logic systems," in *Information Sciences*, 2013, vol. 220, pp. 149–169.

[29] 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, 2010.

[30] J. R. Castro, O. Castillo, P. Melin, A. Rodriguez-diaz, and O. Mendoza, "Universal Approximation of a Class of Interval Type-2 Fuzzy Neural Networks Illustrated with the Case of Non-linear Identification," *Neural Networks*, pp. 1382–1387, 2011.

[31] D. Chakrborty and N. R. Pal, "Integrated feature analysis and fuzzy rule-based system identification in a neuro-fuzzy paradigm.," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 31, pp. 391–400, 2001.

[32] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, 1993.

[33] H. M. Lee, C. M. Chen, J. M. Chen, and Y. L. Jou, "An efficient fuzzy classifier with feature selection based on fuzzy entropy.," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 31, pp. 426–432, 2001.

[34] X. Liu, S. Member, L. Wang, S. Member, and J. Zhang, "Global and Local Structure Preservation for Feature Selection," pp. 1–13, 2013.

[35] R. Silipo and M. R. Berthold, "Input features' impact on fuzzy decision processes.," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 30, pp. 821–834, 2000.

[36] L. Sánchez, M. Rosario Suárez, J. R. Villar, and I. Couso, "Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data," *Int. J. Approx. Reason.*, vol. 49, pp. 607–622, 2008.

[37] D. Chakraborty and N. R. Pal, "Selecting useful groups of features in a connectionist framework.," *IEEE Trans. Neural Netw.*, vol. 19, pp. 381–396, 2008.

[38] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification.," *IEEE Trans. Neural Netw.*, vol. 15, pp. 110–123, 2004.

[39] Y. L. Y. Lin and G. A. . I. Cunningham, "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.*, vol. 3, 1995.

[40] M. Monirul Kabir, M. Monirul Islam, and K. Murase, "A new wrapper feature selection approach using neural network," *Neurocomputing*, vol. 73, pp. 3273–3283, 2010.

[41] N. R. Guo and T. H. S. Li, "Construction of a neuron-fuzzy classification model based on feature-extraction approach," *Expert Syst. Appl.*, vol. 38, pp. 682–691, 2011.

[42] N. R. Pal and S. Saha, "Simultaneous structure identification and fuzzy rule generation for Takagi-Sugeno models.," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 38, pp. 1626–1638, 2008.

[43] G.-D. Wu and P.-H. Huang, "A Maximizing-Discriminability-Based Self-Organizing Fuzzy Network for Classification Problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, 2010.

[44] H. Han and J. Qiao, "A Self-Organizing Fuzzy Neural Network Based on a Growing-and-Pruning Algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 18, 2010.

[45] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, 1999.

[46] G. Leng, T. M. McGinnity, and G. Prasad, "Design for Self-Organizing Fuzzy Neural Networks Based on Genetic Algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 14. pp. 755–766, 2006.

[47] J. Alcalá-Fdez, R. Alcalá, M. J. Gacto, and F. Herrera, "Learning the membership function contexts for mining fuzzy association rules by using genetic algorithms," *Fuzzy Sets Syst.*, vol. 160, pp. 905–921, 2009.

[48] C. H. Tan, K. S. Yap, and H. J. Yap, "Application of genetic algorithm for fuzzy rules optimization on semi expert judgment automation using Pittsburg approach," *Appl. Soft Comput. J.*, vol. 12, pp. 2168–2177, 2012.

[49] H. Zhang, B. Zhang, and F. Wang, "Automatic Fuzzy Rules Generation Using Fuzzy Genetic Algorithm," *2009 Sixth Int. Conf. Fuzzy Syst. Knowl. Discov.*, vol. 6, 2009.

[50] N. Wang, M. J. Er, and X. y. Meng, "A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks," *Neurocomputing*, vol. 72, pp. 3818–3829, 2009.

[51] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, 1998.

[52] S. W. Tung, C. Quek, and C. Guan, "eT2FIS: An Evolving Type-2 Neural Fuzzy Inference System," *Inf. Sci. (Ny).*, vol. 220, pp. 124–148, Jan. 2013.

[53] D. Chakraborty and N. R. Pal, "Selecting useful groups of features in a connectionist framework.," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 381–96, Mar. 2008.

[54] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, 1995.

[55] H. Ishibuchi, T. Murata, and I. . Türkşen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets and Systems*, vol. 89. pp. 135–150, 1997.

[56] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems.," *IEEE Trans. Syst. Man. Cybern. B. Cybern.*, vol. 29, pp. 601–618, 1999.

[57] Y.-C. Chen, N. R. Pal, and I.-F. Chung, "An Integrated Mechanism for Feature Selection and Fuzzy Rule Extraction for Classification," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 4, pp. 683–698, Aug. 2012.

[58] G. Leng, T. M. McGinnity, G. Prasad, "An approach for online extraction of fuzzy rules using a self-organising fuzzy neural network," Fuzzy Sets Syst., vol. 150, no. 2, pp. 211–243, 2005.

[59] A. Staiano, R. Tagliaferri, and W. Pedrycz, "Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering," *Neurocomputing*, vol. 69, pp. 1570–1581, 2006.

[60] Y. Lan, Y. C. Soh, and G. Bin Huang, "Ensemble of online sequential extreme learning machine," *Neurocomputing*, vol. 72. pp. 3391–3395, 2009.

[61] C.-F. Juang, C.-M. Hsiao, and C.-H. Hsu, "Hierarchical Cluster-Based Multispecies Particle-Swarm Optimization for Fuzzy-System Optimization," *IEEE Trans. Fuzzy Syst.*, vol. 18, 2010.

[62] G. M Mendez, L. Leduc-Lezama, R. Colas, G. Murillo-Perez, J. Ramirez-Cuellar, J. J. Lopez, "Modeling and control of the coiling temperature using type-2 fuzzy logic systems," *Ironmaking and Steelmaking*, vol. 37, no. 2, pp. 126-134, 2009.

[63] Gerardo M. Méndez, Rafael Colás, Luis Leduc, Ismael Lopez-Juarez, Rigoberto Longoria, " Finishing mill thread speed setup and control by interval type-1 non-singleton type-2 fuzzy logic systems," *Ironmaking and Steelmaking*, vol. 39, no. 5, pp. 342-354, 2012.