This is the Pre-Published Version.
The following publication Z. Deng, K. Choi, L. Cao and S. Wang, "T2FELA: Type-2 Fuzzy Extreme Learning Algorithm for Fast Training of Interval Type-2 TSK Fuzzy Logic System," in IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 4, pp. 664-676, April 2014 is available at http://dx.doi.org/10.1109/TNNLS.2013.2280171.

T2FELA: Type-2 Fuzzy Extreme Learning Algorithm for Fast Training of Interval Type-2 TSK Fuzzy Logic System

Zhaohong Deng, Kup-Sze Choi, Longbing Cao, Shitong Wang

***Abstract*—A challenge in modeling type-2 fuzzy logic systems is the development of efficient learning algorithms to cope with the ever increasing size of real-world datasets. In this study, the extreme learning strategy is introduced to develop a fast training algorithm for interval type-2 Takagi-Sugeno-Kang fuzzy logic systems (IT2 TSK FLSs). The proposed algorithm, called Type-2 Fuzzy Extreme Learning Algorithm (T2FELA), has two distinctive characteristics. First, the parameters of the antecedents are randomly generated and the parameters of the consequents are obtained by a fast learning method according to the extreme learning mechanism. Moreover, since the obtained parameters are optimal in the sense of minimizing the norm, the resulting fuzzy systems exhibit better generalization performance. The experimental results clearly demonstrate that the training speed of the proposed T2FELA algorithm is superior to that of the existing state-of-the-art algorithms. The proposed algorithm also shows competitive performance in generalization abilities.**

***Index Terms*—Type-2 Fuzzy Logic System, Extreme Learning, Parameter optimization, Fast training.**

I. INTRODUCTION

Fuzzy logic systems (FLSs) have been extensively applied for approximate reasoning and learning in system modeling, intelligent control, signal processing and prediction. FLSs have experienced two important stages of development since 1970: the classical type-1 (T1) FLSs using the T1 fuzzy set and the advanced type-2 (T2) FLSs using the T2 fuzzy set. While T1 FLSs have been comprehensively studied from both the theoretical and application aspects [1-3, 52, 53], T2 FLSs have been attracting more attention in the last decade [4-6, 56-64] due to its superior performance in modeling uncertainty frequently encountered in real-world modeling tasks. In addition, the interpretation and learning abilities of T2 FLSs remain as strong as that of T1 FLSs.

The Interval T2 FLSs (IT2 FLSs) are the most extensively used T2 FLSs because of their efficiency and simplicity. However, IT2 FLSs are facing a challenge that hampers further extensive applications: the training efficiency cannot cope with the increasing size of datasets. For example, if IT2 FLSs are used for biochemical process modeling that involves large datasets, the classical learning algorithms may be inefficient due to high computational cost. This can severely prohibit extensive applications of IT2 FLSs since large datasets are becoming major information sources for IT2 FLS learning and construction in real-world applications, e.g. intelligent control of biochemical processes and market trend analysis. While data-driven learning [7-13] is the most efficient and common method used to deal with rapid and massive accumulation of data in real-world applications, the high computational complexity of the classical data-driven IT2 FLS learning algorithms remain an issue. This brings about a theoretical and practical need to train typical IT2 FLS models efficiently on large datasets.

Research on fast-learning intelligent models has been conducted to improve the performance of existing machine learning methods on handling information sources with large datasets. For example, Tseng et al. studied fast and scalable leaning approaches for kernel methods such as support vector machines (SVMs) [14, 15]. Deng et al. proposed effective approaches for scalable learning of kernel density estimation and classical T1 FLSs [16-18]. Huang et al. [19-21] investigated fast and scalable learning methods for generalized single-hidden layer feed-forward neural networks (SLFNs).

The extreme learning machine (ELM) theory has emerged in recent years as one of the most powerful methods for fast learning of intelligent models [19-25]. ELM is originally proposed for SLFNs [19, 20] and further extended for generalized SLFNs where the hidden layer nodes are not necessarily neuron alike [26, 27]. In ELM, all the hidden node parameters are randomly generated. Huang et al. [19, 20] have proved that SLFNs exhibit universal approximation capability with a wide variety of random computational hidden nodes. Examples of the computational hidden nodes include additive/radial basis function (RBF) hidden nodes, fuzzy rules [28] and wavelets [29].

Training methods based on the ELM strategy in general have the following distinctive advantages [19-30].

First, different from common learning strategies, tuning of the nodes in the hidden layer of SLFNs is not required. Typically, ELM is implemented by employing random computational nodes in the hidden layer, which may be independent of the training data.

Second, unlike the traditional learning algorithms for neural networks (NNs), ELM not only targets to achieve the smallest training error but also the smallest norm of the output weights.

This work was supported in part by the Hong Kong Research Grants Council (PolyU 5134/12E), and the National Natural Science Foundation of China under Grant Nos. 60903100, 60975027, 61170122 and Jiangsu Engineering R&D Center for Information Fusion Software under Grant SR-2011-01.

Z.H. Deng is with the School of Digital Media, Jiangnan University, Wuxi 214122, China and the Centre for Integrative Digital Health, the Hong Kong Polytechnic University (e-mail: dzh666828@yahoo.com.cn).

K.S. Choi is with the Centre for Integrative Digital Health, the Hong Kong Polytechnic University(e-mail: kschoi@ieee.org)

L.B. Cao is with the Advanced Analytics Institute, University of Technology Sydney, Australia (e-mail: longbing.cao@uts.edu.au)

S.T. Wang is with the School of Digital Media, Jiangnan University, Wuxi 214122, China (e-mail: wxwangst@yahoo.com.cn).

Attributed to this feature, feedforward neural networks trained by ELM, according to neural network theory [30], are expected to exhibit better generalization performance.

Third, with ELM, it is not required to tune the nodes in the hidden layer of SLFNs and thus the hidden layer parameters can be fixed. The output weights can then be obtained by only solving the corresponding linear equations, which is very efficient compared to classical methods such as gradient-descent learning based methods (e.g. back-propagation algorithm) and standard optimization methods such as the quadratic programming (QP) based algorithm.

The relationships between FLSs and NNs have been studied extensively [31-35]. It is revealed that under some constraints, many FLSs can be interpreted as specific NNs and trained by using NN learning algorithms. The NNs can also be interpreted with special fuzzy inference rules. For example, several typical type-2 fuzzy NNs (T2 FNNs) [11-12] have been used to develop the learning methods for different T2 FLS models.

Inspired by the distinctive advantages of ELM and the important relationships between T2 FLSs and fuzzy NNs (FNNs) as discussed above, we propose to develop a fast training algorithm for IT2 TSK FLSs by leveraging the learning mechanism of ELM. The proposed training algorithm inherits the characteristics of ELM and has the following advantages for IT2 TSK FLSs training: (1) the parameters of the antecedents can be randomly generated and the parameters of the consequents can be learned quickly; (2) the fuzzy systems obtained have better generalization performance since the parameters of the consequents are optimal in the sense of minimizing the norm.

The contributions of our work are highlighted as follows.

(1) We show that the ELM strategy can be adopted for training IT2 TSK FLSs by revealing that the training of IT2 TSK FLSs can be taken as a special case of the training of the generalized SLFNs.

(2) With the ELM strategy, we propose a novel algorithm T2FELA for training IT2 TSK FLSs from the viewpoint of NN learning.

(3) The proposed algorithm is shown to outperform the existing state-of-the-art algorithms through extensive experiments conducted on synthetic and real-world datasets.

The rest of this paper is organized as follows. Section II discusses the IT2 TSK FLS and the classical ELM. In Section III, the T2FELA algorithm is proposed for fast training of IT2 TSK FLS. The algorithm is evaluated experimentally in Section IV. Finally, a conclusion is made in Section V.

II. Related Work

In this section, we first introduce the typical IT2 TSK FLS. The classical ELM is then briefly described, which will be adopted to develop the proposed T2FELA algorithm for fast learning of IT2 TSK FLSs.

A. IT2 TSK FLS

1) Fuzzy Inference Rules and Inference Mechanism: For traditional T1 TSK FLSs, the most commonly used fuzzy inference rules are defined as follows [1, 2].

Rules of T1 TSK FLS R^k :

If x_1 is A_1^k $\wedge x_2$ is A_2^k $\wedge \cdots \wedge x_d$ is A_d^k

Then $w^k(\mathbf{x}) = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d$, $k = 1, \cdots, K$,

where A_i^k is a T1 fuzzy subset subscribed by the input variable x_i for the *k*-th rule; *K* is the number of fuzzy rules and $\wedge$ is a fuzzy conjunction operator; $\mathbf{p}^k = [p_0^k, p_1^k, \cdots, , p_d^k]^T$ denotes the consequent parameters of the *k*-th fuzzy rule. Each rule is premised on the input vector $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, and maps the fuzzy sets in the input space $A^k \subset R^d$ to a varying singleton denoted by w^k .

Since IT2 TSK FLS is easy to design and implement, it is widely used for T2 FLSs. In IT2 TSK FLSs, the IT2 fuzzy sets are adopted to replace the T1 fuzzy sets in the traditional T1 TSK FLSs. One type of commonly used fuzzy inference rules for IT2 TSK FLSs can be expressed as follows [11],

Rules of IT2 TSK FLS R^k :

If x_1 is $\tilde{A}_1^k$ $\wedge x_2$ is $\tilde{A}_2^k$ $\wedge \cdots \wedge x_d$ is $\tilde{A}_d^k$

Then $w^k = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d$, $k = 1, \cdots, K$,

where $\tilde{A}_i^k$ is an IT2 fuzzy subset subscribed by the input variable x_i for the *k*-th rule; *K* is the number of fuzzy rules and $\wedge$ is a fuzzy conjunction operator; $\mathbf{p}^k = [p_0^k, p_1^k, \cdots, , p_d^k]^T$ denotes the consequent parameters of the *k*-th fuzzy rule. Each rule is premised on the input vector $\mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, and maps the fuzzy sets in the input space $\tilde{A}^k \subset R^d$ to a varying singleton denoted by w^k . For the IT2 fuzzy set $\tilde{A}_i^k$, different types of primary membership function (MF) can be defined. A Gaussian primary membership function with a fixed standard deviation σ_i^k and an uncertain mean that takes on values $[m_i^{k1}, m_i^{k2}]$ is expressed as

$$\mu_{\tilde{A}_i^k}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - m_i^k}{\sigma_i^k}\right)^2\right] \equiv N(x_i; m_i^k, \sigma_i^k), \quad m_i^k \in [m_i^{k1}, m_i^{k2}] \tag{1.a}$$

where m_i^k, σ_i^k denote the means and width parameters of Gaussian membership function, respectively; the value of Eq. (1.a) is in a bounded interval defined by the lower primary MF $\underline{\mu}_{\tilde{A}_i^k}(x_i)$ and upper primary MF $\overline{\mu}_{\tilde{A}_i^k}(x_i)$ as follows.

$$\underline{\mu}_{\tilde{A}_i^k}(x_i) = \begin{cases} N(x_i;\ m_i^{k2}, \sigma_i^k) & x_i \le \dfrac{m_i^{k1} + m_i^{k2}}{2} \\ N(x_i;\ m_i^{k1}, \sigma_i^k) & x_i > \dfrac{m_i^{k1} + m_i^{k2}}{2} \end{cases} \tag{1.b}$$

$$\overline{\mu}_{\tilde{A}_i^k}(x_i) = \begin{cases} N(x_i;\ m_i^{k1}, \sigma_i^k) & x_i < m_i^{k1} \\ 1 & m_i^{k1} \le x_i \le m_i^{k2} \\ N(x_i;\ m_i^{k2}, \sigma_i^k) & x_i > m_i^{k2} \end{cases} \tag{1.c}$$

Each rule performs a fuzzy meet operation using an algebraic product operation. The output of the *if-part* of a rule is a firing strength which is an interval T1 (IT1) fuzzy set as defined by

$$F_k = [\underline{f}_k, \overline{f}_k], \tag{2.a}$$

where

$$\underline{f}_k = \prod_{i=1}^{d} \underline{\mu}_{\tilde{A}_i^k} \tag{2.b}$$

and

$$\overline{f}_k = \prod_{i=1}^{d} \overline{\mu}_{\tilde{A}_i^k} . \tag{2.c}$$

The output of the *then-part* of the *k*-th rule can be expressed as

$$w^k = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d = \sum_{i=0}^{d} p_i^k x_i . \tag{3}$$

By applying a T2 reducer, the IT1 fuzzy set $[y_l, y_r]$ for the model output can be computed as follows.

$$[y_l, y_r] = \int_{w^1} \cdots \int_{w^K} \int_{f_1 \in [\underline{f}_1, \overline{f}_1]} \cdots \int_{f_K \in [\underline{f}_K, \overline{f}_K]} 1 \Big/ \frac{\sum_{k=1}^{K} f_k w^k}{f_k} \tag{4}$$

Since there is no direct theoretical solution for Eq. (4), the computation of the reduced set resorts to iterative methods. One commonly used method is the Karnik-Mendel (K-M) iterative procedure [5, 48], where the consequent values should be re-ordered in an ascending order. Denote the original rule-ordered consequent values as $\mathbf{w} = [w^1, \cdots, w^K]^T$ and the re-ordered sequence as $\tilde{\mathbf{w}} = [\tilde{w}^1, \cdots, \tilde{w}^K]^T$, where $\tilde{w}^1 \le \tilde{w}^2 \le \cdots \le \tilde{w}^K$, the relationship between $\mathbf{w}$ and $\tilde{\mathbf{w}}$, according to [7, 11], is given by

$$\tilde{\mathbf{w}} = \mathbf{Q}\mathbf{w}, \tag{5}$$

where $\mathbf{Q}$ is a $K \times K$ permutation matrix. The elementary vectors in $\mathbf{Q}$ are used as the columns, and they are arranged so that the elements in $\mathbf{w}$ are moved to new locations and arranged in an ascending order in the transformed vector $\tilde{\mathbf{w}}$. For the elementary vectors, all of the elements are zero except one unity element in a specified position. Details about the construction of $\mathbf{Q}$ can be found in [7]. Accordingly, the rule orders $\underline{f}_k$ and $\overline{f}_k$ can be rearranged to yield the re-ordered sequence $\tilde{\underline{f}}_k$ and $\tilde{\overline{f}}_k$ respectively. The outputs y_l and y_r in Eq. (4) can then be computed as follows:

$$y_l = \frac{\sum_{k=1}^{L} \tilde{\overline{f}}_k \tilde{w}^k + \sum_{k=L+1}^{K} \tilde{\underline{f}}_k \tilde{w}^k}{\sum_{k=1}^{L} \tilde{\overline{f}} + \sum_{k=L+1}^{K} \tilde{\underline{f}}_k} \tag{6.a}$$

and

$$y_r = \frac{\sum_{k=1}^{R} \tilde{\underline{f}}_k \tilde{w}^k + \sum_{k=R+1}^{K} \tilde{\overline{f}}_k \tilde{w}^k}{\sum_{k=1}^{R} \tilde{\underline{f}} + \sum_{k=R+1}^{K} \tilde{\overline{f}}}, \tag{6.b}$$

where L and R are the switch points obtained by the K-M algorithm [5, 48] or its modified versions [49, 50]. With the IT1 fuzzy set $[y_l, y_r]$, the final output can be computed by averaging y_l and y_r, i.e.,

$$y = (y_l + y_r)/2 \tag{7}$$

Some modified K-M algorithms have also been proposed to obtain the switch points with reduced computational time [49, 50], e.g. the enhanced K-M algorithm [49]. However, in this study, the basic K-M method [5, 48] is used since it is simple to implement for all the related learning algorithms and sufficient for performance comparison in our experiments.

2) Data-driven Training Algorithms: Data-driven learning has become the most important method for the construction of various IT2 FLSs to model real-world tasks, where the learning of model parameters by using the available data is a critical task. Some effective learning methods have been proposed for IT2 FLSs, including the gradient learning-based method [7], Singular Value Decomposition-QR decomposition (SVD-QR) method [8], dynamical optimal training method [9], support vector learning-based method [11], self-organizing evolving-based methods [12, 13], different mechanisms based hybrid learning algorithm [10, 54, 64, 65-68], and the bio-inspired methods and evolutionary learning based method [55, 59-63]. These algorithms have been used for training typical IT2 FLS models, such as IT2 Mamdani-Larsen FLSs (IT2 ML FLSs) and IT2 TSK FLSs. While several of these methods work efficiently with both small and large datasets, faster algorithms are always expected to keep up with ever-increasing demand in many applications. For example, the iteration rules based methods will become very time consuming and the efficiency deteriorates with the increasing size of the training data. Such situations are indeed commonly encountered in FLS-related enterprise applications. Thus, the development of fast learning methods for typical IT2 FLS models is a very important task.

B. ELM

1) Basic ELM : As one of the most powerful methods for fast learning of intelligent models, ELM has attracted considerable attention in recent years [19-25]. The ELM theory is originally proposed for SLFNs [19, 20] and further extended for generalized SLFNs, where the hidden layer nodes are not necessarily neuron alike [26, 27, 51]. The basic ELM is described below.

First, we begin with a brief description of SLFNs. For a given set of training examples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset R^d \times R^m$ for a multiple-input-multiple-output SLFN, if the outputs of the SLFN, $f_l(\mathbf{x}_j)$, are equal to the sample targets, i.e. $t_{j,l}$, we have

$$\begin{aligned} &f_l(\mathbf{x}_j) = \sum_{i=1}^{M} \beta_{i,l} G(\mathbf{a}_i, b_i, \mathbf{x}) = t_{j,l}, \\ &j = 1, \ldots, N, \quad l = 1, \ldots, m, \end{aligned} \tag{8}$$

where $f_l(\mathbf{x}_j)$ denotes the *l*-th output of SLFN with respect to the input sample $\mathbf{x}_j$; N denotes the number of training samples; M denotes the number of the hidden nodes; m denotes the number of the output nodes; $G(\mathbf{a}_i, b_i, \mathbf{x})$ is an active function of hidden nodes with the parameters $\mathbf{a}_i, b_i$ and $\beta_{i,l}$ are the weights connecting the hidden nodes and the output nodes. Then, Eq. (8) can be expressed in the compact form

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{9}$$

where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_M, b_M, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_M, b_M, \mathbf{x}_N) \end{bmatrix}_{N \times M} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix}_{N \times M}, \tag{10.a}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,m} \\ \vdots & \cdots & \vdots \\ \beta_{M,1} & \cdots & \beta_{M,m} \end{bmatrix}_{M\times m}, \quad (10.b)$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix} = \begin{bmatrix} t_{1,1} & \cdots & t_{1,m} \\ \vdots & \cdots & \vdots \\ t_{N,1} & \cdots & t_{N,m} \end{bmatrix}_{N\times m}. \quad (10.c)$$

Here, $\mathbf{t}^T$ is the transpose of vector $\mathbf{t}$; $\mathbf{H}$ is called the hidden layer output matrix; the *i*-th column of $\mathbf{H}$ is the output vector of the *i*-th hidden node with respect to inputs $\mathbf{x}_i$, $i = 1, \cdots, N$; the *j*-th row of $\mathbf{H}$, i.e. $\mathbf{h}(\mathbf{x}_j)$, is the output vector of the hidden layer with respect to input $\mathbf{x}_j$; $\boldsymbol{\beta}$ is the weight matrix connecting the hidden layer to the output layer; and $\boldsymbol{\beta}_i$ is the weight vector connecting the hidden layer to the *i*-th output node.

With the ELM theory, the parameter learning of the SLFNs is achieved as follows. In Eq. (9), an exact solution is usually not available if the number of training data is larger than the number of hidden nodes. However, it has been proved in theory [18, 19] that, for SLFNs with random hidden nodes, the neural networks exhibit the universal approximation capability where the hidden nodes can be randomly generated independent of the training data. Once the hidden nodes are generated by random, the hidden-layer output matrix $\mathbf{H}$ is readily available given the training data, without the need to tune the nodes. Thus, the training of SLFN by ELM is simply reduced to the solution of the linear system in Eq. (9). Under the constraint of minimum norm least square, i.e.,

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{\beta}\| \text{ and } \min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|, \quad (11)$$

the solution of Eq. (9) is given explicitly [19] with the simple representation

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{+}\mathbf{T}, \quad (12)$$

where $\hat{\boldsymbol{\beta}}$ is the optimal solution of $\boldsymbol{\beta}$ and $\mathbf{H}^{+}$ is the Moore–Penrose generalized inverse of $\mathbf{H}$ [36, 37]. Based on the principle of ELM, an ELM learning algorithm for SLFNs has the following distinctive properties for model training.

(1) ELM is implemented by employing random computational nodes in the hidden layer, which may be independent of the training data.

(2) ELM not only targets to achieve the smallest training error but also the smallest norm of the output weights. According to the neural network theory [30], ELM is expected to exhibit better generalization performance.

(3) Since it is not required to tune the nodes in the hidden layer of SLFNs, the hidden layer parameters can be fixed and the output weights can then be solved simply by solving the corresponding linear equations using Eqs. (11) and (12), which is very efficient when compared to classical methods like gradient-descent learning based methods and standard optimization based methods.

2) Advances of ELM: When compared to conventional techniques, ELM offers better generalization performance at faster learning speed and with minimal human intervention. Because of this attractive feature, considerable research efforts have been devoted to develop many improved versions and variants of ELM. The representative work includes the batch learning mode of ELM [19], fully complex ELM [38, 39], online sequential ELM [40, 41], incremental ELM [20,26,27], the ensemble of ELM [42-44] and the advanced ELM [50]. Recently, the bidirectional ELM for regression problem and the universal approximation of ELM with adaptive growth of hidden nodes were studied in [69] and [70], respectively. Many ELM based applications have also been investigated in literature. For example, ELM has been used for bioinformatics [71] and medical diagnosis [72]. In particular, research has been conducted to leverage the ELM learning strategy for fuzzy system training. In [28], the classical T1 TSK FLS is employed as a special SLFN with the fuzzy inference rules as the hidden nodes, and the fuzzy ELM algorithm is then proposed for the fast learning of the traditional T1 TSK FLS. Motivated by the above advances, the ELM learning strategy is investigated in this study to develop the fast learning algorithm T2FELA for IT2 TSK FLSs.

III. T2FELA

Based on the ELM learning strategy, we present the new fast training algorithm T2FELA for IT2 TSK FLS modeling in this section. The framework of the proposed T2FELA is first described, followed by an overview of the main procedure and a detailed description of the key stages.

A. The Framework of T2FELA

The proposed training method contains three stages, as illustrated in Fig. 1. In Stage 1, according to the ELM learning strategy, the parameters of the antecedents are randomly assigned and then fixed in the subsequent stages. In Stage 2, the parameters of the consequents are initialized by solving the linear system in Eq. (9), where the K-M algorithm is *not* used here to compute the switch points since the initial parameters are not available. In Stage 3, the consequents are further refined by using the K-M algorithm to obtain the switch points and the corresponding linear system in Eq. (9) is solved again to obtain the refined parameters.

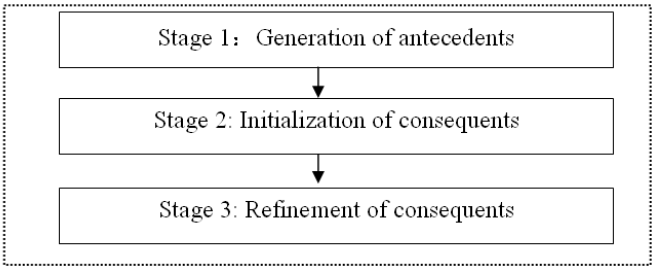


Fig.1 The framework of the proposed ELM strategy based T2FELA.

B. Three Stages of T2FELA

1) Generation of Antecedents: In this stage, Eqs. (1.a)-(1.c) are used to model the fuzzy set of antecedents. The parameters to be determined are the fixed standard deviations σ_i^k and the bounds of the uncertain mean, i.e., $[m_i^{k1}, m_i^{k2}]$. With reference to the related work on the ELM-based methods, these parameters can be randomly assigned within a certain range. In our experiments, all the attributes of the data are normalized into the interval [0, 1]. With the preprocessed data, we

randomly generate the standard deviations σ_i^k within the range [0, 5] and the center $\bar{m}_i^k$ of $[m_i^{k1}, m_i^{k2}]$ within the range [0, 1]. Further, we use $\bar{m}_i^k$ and a fixed minor deviation $\Delta\bar{m}_i^k$ to obtain $m_i^{k1} = \bar{m}_i^k - \Delta\bar{m}_i^k$ and $m_i^{k2} = \bar{m}_i^k + \Delta\bar{m}_i^k$. Note that the above settings cannot assure optimal configuration. While they can be further optimized by some strategies, e.g. the cross-validation strategy, our experimental studies show that the settings are good enough for most situations.

2) Initialization of Consequents: In this stage, the outputs y_l and y_r in Eqs. (6.a) and (6.b) are approximated by the following two equations without using the K-M iterative procedure:

$$y_l = \frac{\sum_{k=1}^{K} \underline{f}_k w^k}{\sum_{k=1}^{K} \underline{f}_k} = \sum_{k=1}^{K} \underline{f}_k' w^k \,,\ \underline{f}_k' = \frac{\underline{f}_k}{\sum_{k'=1}^{K} \underline{f}_{k'}} \tag{13.a}$$

and

$$y_r = \frac{\sum_{k=1}^{K} \overline{f}_k w^k}{\sum_{k=1}^{K} \overline{f}_k} = \sum_{k=1}^{K} \overline{f}_k' w^k \,,\ \overline{f}_k' = \frac{\overline{f}_k}{\sum_{k'=1}^{K} f_{k'}} \,. \tag{13.b}$$

With Eqs. (13.a) and (13.b), Eq. (7) becomes

$$y = \frac{y_l + y_r}{2} = \frac{1}{2}\left(\sum_{k=1}^{K} \underline{f}_k' w^k + \sum_{k=1}^{K} \overline{f}_k' w^k\right). \tag{14}$$

Since $w^k = p_0^k + p_1^k x_1 + \cdots + p_d^k x_d$, we have

$$y_l = \sum_{k=1}^{K} \underline{f}_k' \left(p_0^k + p_1^k x_1 + \cdots + p_d^k x_d \right) \tag{15.a}$$

and

$$y_r = \sum_{k=1}^{K} \overline{f}_k' \left(p_0^k + p_1^k x_1 + \cdots + p_d^k x_d \right), \tag{15.b}$$

let

$$\phi_l(\mathbf{x}) = \left[\underline{f}_1', \underline{f}_1' x_1, \cdots, \underline{f}_1' x_d, \cdots, \underline{f}_K', \underline{f}_K' x_1, \cdots, \underline{f}_K' x_d \right]^T \tag{16.a}$$

$$\phi_r(\mathbf{x}) = \left[\overline{f}_1', \overline{f}_1' x_1, \cdots, \overline{f}_1' x_d, \cdots, \overline{f}_K', \overline{f}_K' x_1, \cdots, \overline{f}_K' x_d \right]^T \tag{16.b}$$

$$\phi(\mathbf{x}) = \frac{1}{2}\left(\phi_l(\mathbf{x}) + \phi_r(\mathbf{x})\right)$$

$$= \frac{1}{2}\Big[\left(\underline{f}_1' + \overline{f}_1'\right), \left(\underline{f}_1' + \overline{f}_1'\right)_1 x_1, \cdots, \left(\underline{f}_1' + \overline{f}_1'\right) x_d, \cdots, \tag{16.c}$$

$$\left(\underline{f}_K' + \overline{f}_K'\right), \left(\underline{f}_K' + \overline{f}'\right) x_1, \cdots, \left(\underline{f}_K' + \overline{f}'\right) x_d \Big] \in R^{(d+1)K}$$

and

$$\mathbf{p} = \left[p_0^1, \cdots, p_d^1, \cdots, p_0^K, \cdots, p_d^K \right]^T \in R^{(d+1)K} \,, \tag{17}$$

Eq. (14) can then be expressed by the linear system

$$y = \phi(\mathbf{x})^T \mathbf{p} \,. \tag{18}$$

For a given training dataset $D = \{\mathbf{x}_i, y_i\}$ of a regression task, when the antecedents of the IT2 TSK FLS are fixed, we can construct the dataset $\tilde{D} = \{\phi(\mathbf{x}_i), y_i\}$ to train the linear system in Eq. (18), where $\phi(\mathbf{x}_i)$ is generated using Eq. (16.c).

Let

$$\mathbf{\Phi}_1 = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{bmatrix} \in R^{N \times K(d+1)} \tag{19.a}$$

and

$$\mathbf{y} = [y_1, \cdots, y_N]^T \,, \tag{19.b}$$

the linear system in Eq. (18) can then be formulated as

$$\mathbf{\Phi}_1 \mathbf{p} = \mathbf{y} \,. \tag{20}$$

The solution of this system can be obtained in a way similar to the approach used in the basic ELM, as shown in Eqs. (11) and (12). That is, by optimizing the following objective

$$\min_{\mathbf{p}} \|\mathbf{p}\| \ \text{ and } \ \min_{\mathbf{p}} \|\mathbf{\Phi}_1 \mathbf{p} - \mathbf{y}\| \,, \tag{21.a}$$

we have

$$\hat{\mathbf{p}} = \mathbf{\Phi}_1^+ \mathbf{y} \,, \tag{21.b}$$

where $\hat{\mathbf{p}}$ is the optimal solution of $\mathbf{p}$ and $\mathbf{\Phi}_1^+$ is the Moore–Penrose generalized inverse of $\mathbf{\Phi}_1$ [19, 36, 37].

3) Refinement of Consequents: With the initial values obtained in Stage 2, the final consequent parameters p_i^k are determined in this stage. Since the values of p_i^k are now available for computing the consequent values w^k for all the rules, the K-M iterative procedure can be used to obtain the output of the IT2 TSK FLS accurately. Let $\mathbf{w} = [w^1, \cdots, w^K]^T$, $\underline{\mathbf{f}} = [\underline{f}_1, \cdots, \underline{f}_K]^T$, $\overline{\mathbf{f}} = [\overline{f}_1, \cdots, \overline{f}_K]^T$, where the firing strengths are expressed according to the original rule order. According to Eq.(5), the relationship between the original rule-ordered consequent values $\mathbf{w}$ and the re-ordered sequence $\tilde{\mathbf{w}}$ ($\tilde{w}^1 \le \tilde{w}^2 \le \cdots \le \tilde{w}^K$) can be expressed as $\tilde{\mathbf{w}} = \mathbf{Q}\mathbf{w}$, where $\mathbf{Q}$ is a $K \times K$ permutation matrix. Then, $\underline{\mathbf{f}}$ and $\overline{\mathbf{f}}$ are rearranged to yield the re-ordered sequence $\tilde{\underline{\mathbf{f}}} = [\tilde{\underline{f}}_1, \cdots, \tilde{\underline{f}}_K]^T$ and $\tilde{\overline{\mathbf{f}}} = [\tilde{\overline{f}}_1, \cdots, \tilde{\overline{f}}_K]^T$, with $\tilde{\underline{\mathbf{f}}} = \mathbf{Q}\underline{\mathbf{f}}$ and $\tilde{\overline{\mathbf{f}}} = \mathbf{Q}\overline{\mathbf{f}}$. Therefore, in Eq. 6(a), $\sum_{k=1}^{L} \tilde{\overline{f}}_k \tilde{w}^k$, $\sum_{k=L+1}^{K} \tilde{\underline{f}}_k \tilde{w}^k$, $\sum_{k=1}^{L} \tilde{\overline{f}}$ and $\sum_{k=L+1}^{K} \tilde{\underline{f}}_k$ can be written in a compact matrix form as $\overline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_1^T \mathbf{E}_1^T \mathbf{Q}\mathbf{w}$, $\underline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_2^T \mathbf{E}_2^T \mathbf{Q}\mathbf{w}$, $\sum_{k=1}^{L} (\mathbf{Q}\overline{\mathbf{f}})_k$ and $\sum_{k=L+1}^{K} (\mathbf{Q}\underline{\mathbf{f}})_k$ respectively. Finally, Eq. (6.a) can be re-expressed in the rule-ordered form

$$y_l = \frac{\overline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_1^T \mathbf{E}_1^T \mathbf{Q}\mathbf{w} + \underline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_2^T \mathbf{E}_2^T \mathbf{Q}\mathbf{w}}{\sum_{k=1}^{L} (\mathbf{Q}\overline{\mathbf{f}})_k + \sum_{k=L+1}^{K} (\mathbf{Q}\underline{\mathbf{f}})_k} = \boldsymbol{\psi}_l^T \mathbf{w} \,, \tag{22.a}$$

where

$$\boldsymbol{\psi}_l^T = [\psi_{l,1}, \cdots, \psi_{l,K}] = \frac{\overline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_1^T \mathbf{E}_1^T \mathbf{Q} + \underline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_2^T \mathbf{E}_2^T \mathbf{Q}}{\sum_{k=1}^{L} (\mathbf{Q}\overline{\mathbf{f}})_k + \sum_{k=L+1}^{K} (\mathbf{Q}\underline{\mathbf{f}})_k} \in R^K \,, \tag{22.b}$$

$$\mathbf{E}_1 = \left[\mathbf{e}_1, \cdots, \mathbf{e}_L, \mathbf{0}, \cdots, \mathbf{0}\right] \in R^{L \times K} \,, \tag{22.c}$$

$$\mathbf{E}_2 = \left[\mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\varepsilon}_1, \cdots, \boldsymbol{\varepsilon}_{K-L}\right] \in R^{(K-L) \times K} \,, \tag{22.d}$$

with $\mathbf{e}_i \in R^L$ $(i = 1, \cdots, L)$ and $\boldsymbol{\varepsilon}_i \in R^{(K-L)}$ $(i = 1, \cdots, K-L)$ as the elementary vectors, i.e., all the elements are equal to 0 except for the unity *i*-th element.

Similarly, Eq. (6.b) can be re-expressed in the rule-ordered form

$$y_r = \frac{\underline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_3^T \mathbf{E}_3^T \mathbf{Q}\mathbf{w} + \overline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_4^T \mathbf{E}_4^T \mathbf{Q}\mathbf{w}}{\sum_{k=1}^{L} (\mathbf{Q}\underline{\mathbf{f}})_k + \sum_{k=L+1}^{K} (\mathbf{Q}\overline{\mathbf{f}})_k} = \boldsymbol{\psi}_r^T \mathbf{w} \,, \tag{23.a}$$

where

$$\boldsymbol{\psi}_r^T = [\psi_{r,1}, \cdots, \psi_{r,K}] = \frac{\underline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_3^T \mathbf{E}_3^T \mathbf{Q} + \overline{\mathbf{f}}^T \mathbf{Q}^T \mathbf{E}_4^T \mathbf{E}_4^T \mathbf{Q}}{\sum_{k=1}^{L} (\mathbf{Q}\underline{\mathbf{f}})_k + \sum_{k=L+1}^{K} (\mathbf{Q}\overline{\mathbf{f}})_k} \in R^K \,, \tag{23.b}$$

$$\mathbf{E}_3 = [\mathbf{e}_1, \cdots, \mathbf{e}_R, \mathbf{0}, \cdots, \mathbf{0}] \in R^{R\times K}, \quad (23.c)$$

$$\mathbf{E}_4 = [\mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\varepsilon}_1, \cdots, \boldsymbol{\varepsilon}_{K-R}] \in R^{(K-R)\times K}, \quad (23.d)$$

with $\mathbf{e}_i \in R^R$ $(i = 1, \cdots, R)$ and $\boldsymbol{\varepsilon}_i \in R^{(K-R)}$ $(i = 1, \cdots, K-R)$ as the elementary vectors. Based on Eqs. (22.a) and (23.a), Eq. (7) becomes

$$\begin{aligned} y = \frac{y_l + y_r}{2} &= \frac{1}{2}(\boldsymbol{\psi}_l^T + \boldsymbol{\psi}_r^T)\mathbf{w} = \frac{1}{2}\sum_{k=1}^{K}(\boldsymbol{\psi}_l + \boldsymbol{\psi}_r)_k w^k \\ &= \frac{1}{2}\sum_{k=1}^{K}(\boldsymbol{\psi}_l + \boldsymbol{\psi}_r)_k \left(\sum_{j=0}^{d} x_j p_j^k\right), \; x_0 \equiv 1. \end{aligned} \quad (24)$$

Furthermore, let

$$\begin{aligned} \tilde{\phi}(\mathbf{x}) = \frac{1}{2}\big[&(\psi_{l,1} + \psi_{r,1})x_0, \cdots, (\psi_{l,1} + \psi_{r,1})x_d, \cdots, \\ &(\psi_{l,K} + \psi_{r,K})x_0, \cdots, (\psi_{l,K} + \psi_{r,K})x_d\big]^T \in R^{K(d+1)} \end{aligned} \quad (25)$$

and with $\mathbf{p}$ defined in Eq. (17), Eq. (24) can be expressed as the linear system

$$y = \tilde{\phi}(\mathbf{x})^T \mathbf{p}. \quad (26)$$

By denoting $\boldsymbol{\Phi}_2 = \begin{bmatrix} \tilde{\phi}(\mathbf{x}_i)^T \\ \vdots \\ \tilde{\phi}(\mathbf{x}_N)^T \end{bmatrix} \in R^{N\times K(d+1)}$, Eq. (26) is rewritten as

$$\boldsymbol{\Phi}_2 \mathbf{p} = \mathbf{y}. \quad (27)$$

where $\mathbf{y}$ is defined in Eq. (19.b). The solution of the linear system in Eq. (27) can be obtained by optimizing the following objective

$$\min_{\mathbf{p}} \|\mathbf{p}\| \text{ and } \min_{\mathbf{p}} \|\boldsymbol{\Phi}_2 \mathbf{p} - \mathbf{y}\|. \quad (28.a)$$

We then have

$$\hat{\mathbf{p}} = \boldsymbol{\Phi}_2^+ \mathbf{y}, \quad (28.b)$$

where $\hat{\mathbf{p}}$ is the optimal solution of $\mathbf{p}$ and $\boldsymbol{\Phi}_2^+$ is the Moore–Penrose generalized inverse of $\boldsymbol{\Phi}_2$ [19, 36, 37]. The result is also similar to that in Eqs. (11) and (12) in the basic ELM.

C. The T2FELA Algorithm

Based on the three-stage training method discussed above, the corresponding T2FELA algorithm is given as follows.

Algorithm: T2FELA

Stage 1:	Randomly assign the parameters of the antecedents according to the ELM mechanism.
Stage 2:	Initialize the parameters of the consequents using Eqs. (13)-(21).
Stage 3:	Refine the parameters of the consequents using Eqs. (22)- (28).

Remark 1: Different from some classical algorithms, the proposed algorithm is not an iterative procedure between antecedent learning and consequent learning. The parameters in the antecedents are randomly generated only *once* according to the extreme learning theory. Then by optimizing the parameters of the consequents, the obtained IT2 TSK FLS can still be a universal approximator as proved in [19,20] from the viewpoint of SLFN learning. In particular, similar learning procedure is also used in the existing interval type-2 fuzzy neural network with support-vector regression (IT2FNN-SVR) algorithm [11], which first generates the antecedents by a self-evolving strategy and then learns the consequents using a the two-stage SVR with the antecedents fixed. However, the computational time of IT2FNN-SVR remains high since it is usually necessary to solve the QP problem in SVR.

Remark 2: The proposed T2FELA algorithm can be regarded as a special case of ELM learning where the hidden nodes of the SLFN involved are a set of T2 fuzzy inference rules. Thus, the proposed algorithm inherits the virtues of ELM learning strategy as described below.

(1) The learning speed is expected to be much faster since no iterations are involved. This is a characteristic that has been demonstrated by most existing ELM based algorithms [19-25, 51].

(2) The IT2 TSK FLSs obtained by using the proposed algorithm is expected to have better generalization ability since the parameters of the consequents are optimal in the sense of minimizing the norm, as shown in Eqs. (21.a) and (28.a), which can be easily solved by using the Moore–Penrose generalized inverse as in other ELM based algorithms [19-25, 51].

Remark 3: Although the proposed T2FELA algorithm has distinctive advantages in computational complexity, it has an inherent drawback due to the learning mechanism. Since the parameters of the antecedents are generated randomly in the ELM strategy, the ability to interpret the rules may be reduced. Although the input space of the antecedents is partitioned more randomly, we can still give an interpretation with the induced inference rules. Of course, we can improve the interpretation by using other partition strategies to replace the random generation approach, e.g. clustering techniques, but it will inevitably increase the computational cost. Nevertheless, as the classical T1 FLSs using the ELM strategy [28], the IT2 TSK FLS obtained with random generation of the parameters in the antecedents remains a universal approximator. Thus the IT2 TSK FLSs trained by the proposed T2FELA algorithm is expected to possess good approximation ability.

IV. Evaluation

The performance of the proposed T2FELA algorithm is evaluated by comparing with that of the three existing IT2 FLS/FNN training algorithms on synthetic and real-world datasets.

A. Experimental Settings

1) Baseline Algorithms for Performance Comparison: Three algorithms are used for comparison with the proposed T2FELA algorithm, namely, (a) gradient-learning based algorithm (GL-IT2FLS) [7], (b) IT2FNN-SVR [11], and (c) self-evolving interval type-2 fuzzy neural network (SEIT2FNN) [12]. The implementation of these algorithms and our algorithm T2FELA are described below.

GL-IT2FLS: All the parameters of both the antecedents and consequents of the IT2 FLSs are adjusted iteratively by the gradient-descent learning rules.

SEIT2FNN: The parameters of the antecedents of the IT2 FNNs are initialized by the self-evolving strategy. The

parameters of the antecedents and consequents are then adjusted iteratively by the gradient-descent learning rules and the rule-ordered Kalman filtering algorithm respectively.

IT2FNN-SVR: The parameters of the antecedents of the IT2 FNNs are determined by the self-evolving strategy, while the parameters of the consequents are determined by the two-phase SVR learning strategy.

T2FELA: the parameters of antecedents of the IT2 TSK FLSs are randomly generated according to the extreme learning theory, while the parameters of the consequents are determined by the two-phase extreme learning strategy.

2) Parameter Setting: For all the algorithms, the hyper parameters are determined by applying the cross-validation strategy on the training sets. For the proposed T2FELA algorithm and GL-IT2FLS [7], the hyper parameter is the number of fuzzy rules and the optimal value is determined by the cross-validation strategy within the parameter set $\{4,9,16,25,36,49,64,81,100,121\}$. For the algorithms IT2FNN-SVR [11] and SEIT2FNN [12], the hyper parameter is the threshold of the firing-strength, which directly influences the final number of fuzzy rules of the generated T2 FLSs. It is determined by the cross-validation strategy within the parameter set $\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8\}$. For the algorithm IT2FNN-SVR [11], the hyper parameter C, i.e, the regularization parameter in SVR, is determined by the cross-validation strategy within the parameter set $\{2^{-5},2^{-4},\ldots,2^{4},2^{5}\}$. For the iteration based algorithms, i.e., GL-IT2FLS [7] and SEIT2FNN [12], the maximum iteration number is set to be 10.

3) Evaluation Indices: In all the experiments, the main performance index is the training time T of the algorithms.

Furthermore, the performance index J as defined in Eq. (29) is also adopted [2, 17, 18, 35] to evaluate the generalization abilities of the algorithms on the testing datasets:

$$J=\sqrt{\frac{\sum_{i=1}^{N}(y_i'-y_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}}\,, \tag{29}$$

where N is the number of test data; y_i is the output for the i-th test input; y_i' is the fuzzy model output for the i-th test input and $\bar{y}=\sum_{i=1}^{N}y_i\big/N$. The smaller the value of J, the better the generalization performance.

4) Other Settings: For all the algorithms, the classical K-M algorithm [5, 48] is adopted for the computation of the switch points in Eqs. (6.a) and (6.b) for simplicity. Although other faster algorithms can also be adopted [49, 50], it is considered a fair approach to use the same K-M algorithm for performance comparison for all the training algorithms in our experiments.

All the algorithms in the experiments are implemented using Matlab. For the IT2FNN-SVR algorithm, the LibSVM code is used to solve the corresponding SVR [47]. The experiments are conducted on a computer with an Intel Core 2 Duo 2.00 GHz CPU and 2GB RAM. The experimental settings are summarized in Table I.

B. Datasets

Three types of datasets are adopted for performance evaluation, including synthetic datasets, benchmarking real-world datasets, and biochemical process modeling datasets. For all the datasets, the data attributes are normalized into the range [0, 1]. The details of these datasets and the corresponding experiment results are discussed in Sections IV-C-1 through IV-C-3.

C. Results and Discussions

1) Synthetic Datasets: The experiments conducted with the synthetic *Friedman* datasets [45] are reported in this section. For these datasets, the input attributes $\mathbf{x}=(x_1,x_2,\cdots,x_5)^T$ are generated independently, each of which is uniformly distributed over [0, 1]. The target is defined by

$$y=10\sin(\pi x_1x_2)+20(x_3-0.5)^2+10x_4+5x_5+\sigma(0,1)\,, \tag{30}$$

where $\sigma(0,1)$ is a noise term which is normally distributed with mean 0 and variance 1.

TABLE I
THE EXPERIMENTAL SETTINGS

Settings	Algorithms			
	SEIT2FNN [12]	IT2FNN-SVR [11]	GL-IT2FLS [7]	T2FELA
Hyper parameters	The number of fuzzy rules: its optimal value is determined within the parameter set $\{4,9,16,25,36,49,64,81,100,121\}$ by the cross-validation strategy.		The firing-strength: its optimal value is determined within the parameter set $\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8\}$ by the cross-validation strategy.	
		The regularization parameter in SVR: it is determined within the parameter set $\{2^{-5},2^{-4},\ldots,2^{4},2^{5}\}$ by the cross-validation strategy.		
Evaluation indices	1) T: training time on the training dataset 2) J: generalization performance on the testing dataset			
Others	1) The classical K-M algorithm [5, 48] is adopted for the computation of the switch points in Eqs. (6.a) and (6.b). 2) The LibSVM code [47] is adopted to solve the corresponding SVR in the algorithm IT2FNN-SVR. 3) For the iteration based algorithms, i.e., the GL-IT2FLS and SET2FNN, the maximum iteration number is set to be 10.			

In the experiments, datasets of different sizes are generated and used for the training. The size of the generated training sets is in the range of 100 to 5×10^4. Meanwhile, a noise-free dataset of size 10^3 is generated for testing. The performance of each of the four IT2 FLS/FNN training algorithms is then evaluated using these datasets. Each experiment is repeated 20 times for datasets at each size to obtain the average evaluation indices and the corresponding standard deviations.

The training time T and the generalization performance index J of the four training algorithms are shown in Tables II and III respectively. The average values are also compared in Fig. 2.

The number of fuzzy rules given in Table II is determined by performing the cross-validation strategy on the training sets.

Note that for IT2FNN-SVR [11] and SEIT2FNN [12], the parameter determined by the cross-validation strategy is the threshold of firing-strength, which has important influence on the number of fuzzy rules generated by these two algorithms. The smaller the threshold of firing-strength, the smaller the number of the generated fuzzy rules. The following observations can be made from the experimental results:

(1) The efficiency of the proposed T2FELA algorithm is better than that of the other three algorithms. In these algorithms, the training time depends on the learning strategy, the size of training set, and the number of fuzzy rules. For the proposed T2FELA algorithm, while the optimal number of fuzzy rules obtained by the cross-validation strategy is equivalent to those obtained in the other three algorithms, the training time is obviously the smallest.

(2) The generalization ability of the proposed T2FELA algorithm on the *Friedman* datasets is highly competitive among the training algorithms. Although the IT2FNN-SVR algorithm has shown better generalization performance for small datasets, the T2FELA is a more promising algorithm for handling large training sets.

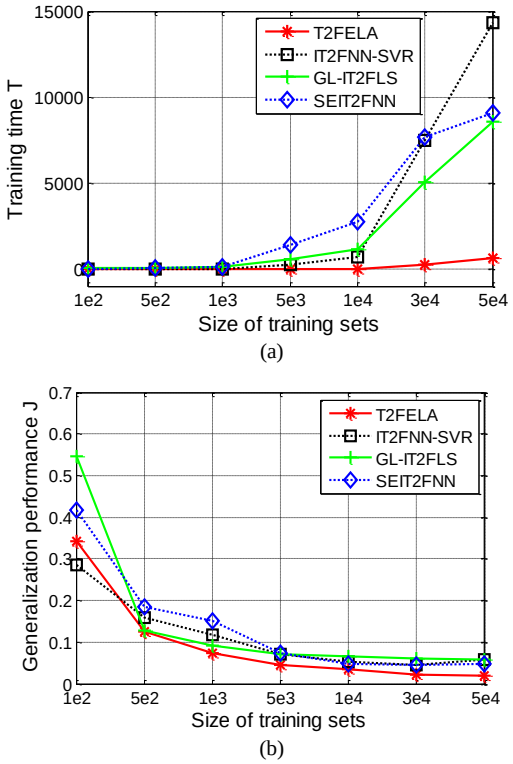


Fig.2 Performance of the four IT2 FLS/FNN training algorithms on Friedman synthetic datasets: (a) training time (seconds) *T*; (b) generalization performance index *J*.

Table II

TRAINING TIME (SECONDS) OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON FRIEDMAN SYNTHETIC DATASETS

Indices	*N*	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
T	1e2	0.05	0.62	32.64	4.00
		0.01	0.12	2.09	0.31
		(16)	(29)	(49)	(5)
	5e2	0.25	3.40	56.74	55.50
		0.032	0.62	1.31	0.87
		(16)	(21)	(16)	(20)
	1e3	0.50	17.64	111.26	129.75
		0.04	1.33	2.77	1.04
		(25)	(48.7)	(16)	(21.5)
	5e3	6.89	269.39	561.60	1395
		0.22	16.10	7.8	2.32
		(25)	(68.33)	(16)	(46.5)
	1e4	22.99	721.22	1148	2716
		0.12	26.31	54.78	2.09
		(36)	(53.67)	(16)	(47.5)
	3e4	240.86	7469	5038	7667
		1.58	24.20	86	198
		(36)	(35.5)	(25)	(44.5)
	5e4	643.22	14304	8553	9096
		1.92	391	138	1.39
		(49)	(17)	(25)	(33)

N denotes the size of training sets and the values inside brackets are the average number of fuzzy rules determined by applying the cross-validation strategy on the training sets for 20 times.

TABLE III

GENERALIZATION PERFORMANCE OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON FRIEDMAN SYNTHETIC DATASETS

Indices	*N*	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
J	1e2	0.3418	0.2853	0.5471	0.4172
		0.1255	0.0261	0.0344	0.0727
	5e2	0.1255	0.1592	0.1277	0.1835
		0.0166	0.0261	0.0014	0.0136
	1e3	0.0743	0.1163	0.0915	0.1502
		0.0094	0.0068	0.0077	0.0394
	5e3	0.0445	0.0706	0.0703	0.0727
		0.0013	0.0018	0.0004	0.0079
	1e4	0.0339	0.0519	0.0648	0.0484
		0.0025	0.0015	0.0038	0.0043
	3e4	0.0219	0.0458	0.0606	0.0442
		0.0007	0.0099	0.0020	0.0165
	5e4	0.0205	0.0594	0.0584	0.0480
		0.0048	0.0172	0.0007	0.0225

N denotes the size of training sets.

2) Real-world Datasets: In this section, the performance of the four algorithms is evaluated using five benchmarking real-world datasets available from the Laboratory of Artificial Intelligence and Computer Science at the University of Porto, Portugal [46]. The datasets are described in Table IV. In the experiments, each dataset is randomly partitioned with the ratio of 4:1 for training and test respectively. This procedure is repeated 20 times to obtain the average performance of each algorithm on each of the five real-world datasets.

The experimental results are shown in Tables V, VI and Fig.3. Similar to the findings obtained in the synthetic datasets, it is noted that the training time of the proposed T2FELA algorithm is obviously less than that of the other three algorithms and the generalization ability is also highly competitive.

On the other hand, we also find from the results that the

training time of the three existing IT2 FLS/FNN training algorithms is at least 10 times slower than that of the proposed algorithm on the five real-world datasets. Even for the CartExample dataset whose size is the largest among the five datasets, the training time of T2FELA is only about 480 seconds. This suggests that the proposed algorithm is promising for modeling large real-world data sets.

3) Biochemical Process Modeling Datasets: Further experiments are also conducted to evaluate the performance of the four algorithms on the modeling of a biochemical process which involves large datasets [18]. A multiple-input-multiple-output dataset originated from the glutamic acid ferment process is adopted. The input variables of the dataset include ferment time k, glucose concentration S(k), glutamic acid concentration P(k), thalli concentration X(k), stirring speed R(k), and ventilation Q(k) at time k. The output variables are glucose concentration S(k+1), glutamic acid concentration P(k+1) and thalli concentration X(k+1) at time k+1. The IT2 TSK FLS based estimation model is illustrated in Fig. 4. In the experiments, the original data is collected from 1.1×10^4 batches of ferment process. Training sets of different sizes (from 10^2 to 5×10^4) are obtained from the original dataset to train the systems. A set of data with a size of 10^3 is also obtained for testing purposes. The training procedure is repeated 20 times for datasets at each size, and the average performance is recorded for comparison.

TABLE IV
FIVE REAL-WORLD REGRESSION DATASETS

Dataset	Number of samples (Ratio between the training data and testing data)	Number of attributes (Input variables + output variables)
delta_elevators	9516 (4:1)	6+1
Census_8	22784 (4:1)	8+1
CartExample	40768 (4:1)	10+1
cadata	20640 (4:1)	9+1
bank32NH	4499 (4:1)	32+1

TABLE V
TRAINING TIME (SECONDS) OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON REAL-WORLD DATASETS

Indices	Dataset	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
T	delta_elevators	50.37 0.10 (16)	664.13 102.42 (80)	1567 58.53 (16)	1276 47.24 (12.5)
	Census_8	140.43 5.48 (6)	1076 28.04 (12)	4750 40.18 (16)	2671 22.64 (8.5)
	CartExample	481.87 24.2 (16)	30994 567 (25.5)	15945 237 (25)	17156 146 (27.5)
	cadata	448.83 10.67 (25)	805.41 23.90 (14.5)	4265 34.23 (16)	7831 13.49 (29.5)
	bank32NH	34.05 1.16 (9)	242.92 18.75 (22)	1893 37.50 (16)	865.62 16.67 (4)

The values inside brackets are the average number of fuzzy rules determined by applying the cross-validation strategy on the training set for 20 times.

TABLE VI
GENERALIZATION PERFORMANCE OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON REAL-WORLD DATASETS

Indices	Dataset	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
J	delta_elevators	0.5947 0.004	0.6008 0.0002	0.5919 0.0010	0.6042 0.0051
	Census_8	0.6769 0.0307	0.6690 0.0072	0.6348 0.0735	0.6983 0.0383
	CartExample	0.2276 0.0001	0.2294 0.0013	0.2266 0.0013	0.4563 0.0617
	cadata	0.5252 0.0384	0.5510 0.0084	0.5568 0.0431	0.6559 0.0032
	bank32NH	0.7185 0.0340	0.7213 0.0193	0.7211 0.0062	0.7053 0.0124

Tables VII, VIII and Fig. 5 show the performance of the four IT2 FLS/FNN training algorithms on glucose concentration prediction. Furthermore, Figs. 6 and 7 plot the training time and performance of glutamic acid concentration prediction and thalli concentration prediction respectively (note that the results of glutamic acid and thalli concentration prediction are not tabulated here due to space limit). The experimental results on the prediction of these three biochemical variables show that the training time of the proposed T2FELA algorithm is much shorter than that of the other three algorithms. The generalization ability of the proposed algorithm is also highly competitive. In particular, the T2FELA algorithm demonstrates more promising generalization performance for the prediction of glutamic acid concentration and glucose concentration. The proposed algorithm is therefore effective for the biochemical process modelling that involves large datasets.

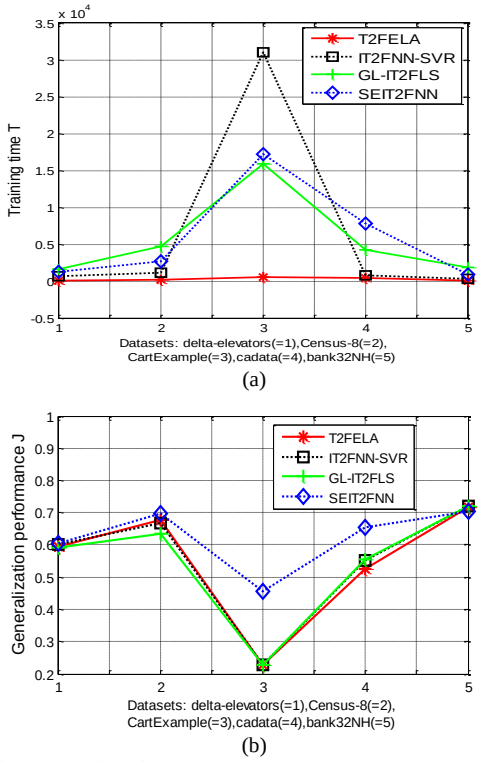


Fig.3 Performance of the four IT2 FLS/FNN training algorithms on five benchmarking real-world datasets: (a) training time (seconds) *T*; (b) generalization performance index *J*.

TABLE VII
TRAINING TIME (SECONDS) OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON BIOCHEMICAL DATASETS FOR GLUCOSE CONCENTRATION (S) PREDICTION

Indices	N	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
T	1e2	0.11	0.09	6.08	1.42
		0.07	0.02	0.39	0.33
		(4)	(3.5)	(4)	(5)
	5e2	0.35	1.28	20.47	9.03
		0.02	0.16	3.27	0.49
		(16)	(18)	(4)	(8.5)
	1e3	0.92	7.95	126.36	24.19
		0.11	3.00	8.65	0.63
		(25)	(45.5)	(16)	(12)
	5e3	20.86	91.15	1763	145.99
		0.31	9.89	31.35	8.03
		(36)	(46.5)	(49)	(15)
	1e4	74.19	348.73	2502	1040
		1.43	13.77	77.06	59.28
		(36)	(51.5)	(36)	(10.5)
	3e4	594.11	1744	7657	3021
		11.03	46.41	185.64	53.82
		(36)	(20.5)	(36)	(10)
	5e4	1587	5786	15880	3855
		67.72	87.68	258.96	45.24
		(36)	(14.5)	(36)	(7.5)

N denotes the size of training sets and the values inside brackets are the average number of fuzzy rules determined by applying the cross-validation strategy on the training sets for 20 times.

TABLE VIII
GENERALIZATION PERFORMANCE OF THE FOUR IT2 FLS/FNN TRAINING ALGORITHMS ON BIOCHEMICAL DATASETS FOR GLUCOSE CONCENTRATION (S) PREDICTION

Indices	N	T2FELA	IT2FNN-SVR	GL-IT2FLS	SEIT2FNN
		Mean/std	Mean/std	Mean/std	Mean/std
J	1e2	0.1197	0.1022	0.1240	0.1025
		0.0325	0.0036	0.0050	0.0065
	5e2	0.0819	0.0818	0.0898	0.0815
		0.0053	0.0012	0.0079	0.0045
	1e3	0.0686	0.0748	0.0807	0.0760
		0.0003	0.0046	0.0087	0.0033
	5e3	0.0561	0.0738	0.0617	0.0707
		0.0023	0.0011	0.0021	0.0075
	1e4	0.0563	0.0667	0.0673	0.0739
		0.0014	0.0025	0.0048	0.0009
	3e4	0.0602	0.0717	0.0665	0.0766
		0.0027	0.0036	0.0045	0.0010
	5e4	0.0562	0.0702	0.0658	0.0789
		0.1197	0.0019	0.0017	0.0029

N denotes the size of training sets.

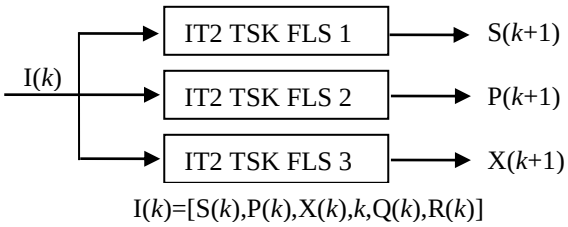
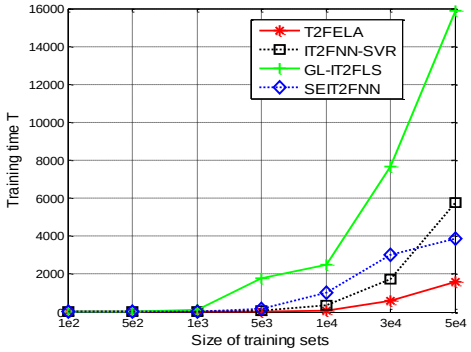
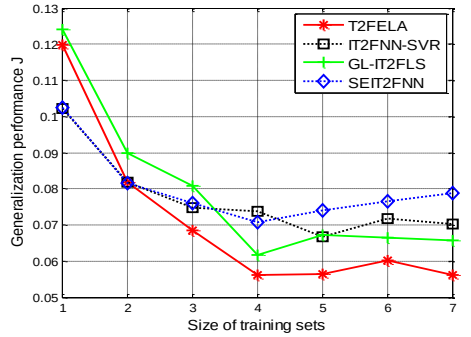


Fig.4 Illustration of the glutamic acid ferment process prediction model based on the IT2 TSK FLSs.



(a)



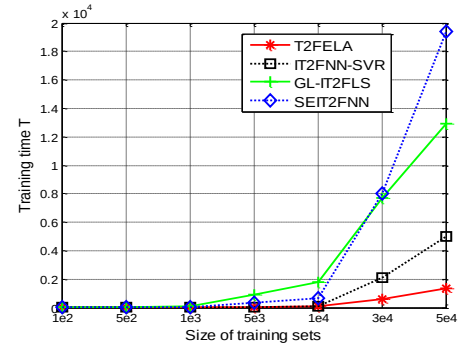
(b)

Fig.5 Performance of the four IT2 FLS/FNN training algorithms on biochemical datasets for glucose concentration (S) prediction: (a) training time (seconds) T; (b) generalization performance index J.

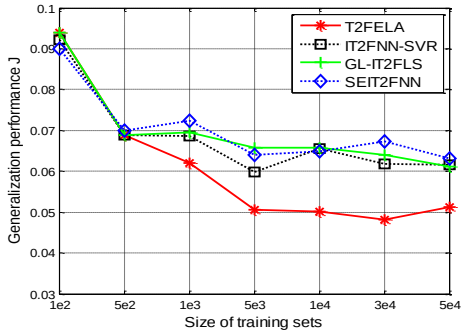
V. CONCLUSIONS

In this study, an extreme learning strategy-based fast training algorithm T2FELA is proposed for the training of IT2 TSK FLSs. It aims at developing a training algorithm that enables the models to be effectively trained on large datasets. Experimental results demonstrate that with the extreme learning mechanism, the proposed T2FELA algorithm allows for random generation of the parameters of the antecedents and fast learning of the parameters of the consequents. Moreover, the resulting IT2 TSK FLSs demonstrate highly competitive generalization performance among several existing algorithms.

While the proposed T2FELA algorithm has demonstrated promising performance for training IT2 TSK FLS on large datasets, there are still rooms for further improvement. For example, the efficiency of the current version of T2FELA algorithm is limited by the demanding memory requirement of the basic ELM learning strategy. In this case, online sequential ELM learning strategy, where data is read chunk by chunk, can be adopted to overcome the difficulties caused by the enormous amount of memory required for handling very large datasets. This will be our future work. Besides, further investigation will be conducted to extend the T2FELA algorithm for other types of T2 FLSs, such as the IT2 ML FLSs.

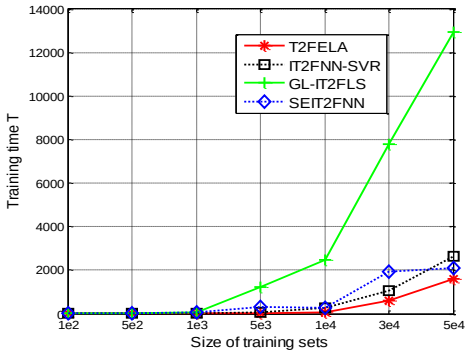


(a)

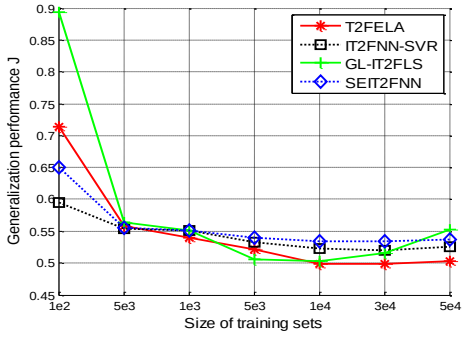


(b)

Fig.6 Performance of the four IT2 FLS/FNN training algorithms on biochemical datasets for glutamic acid concentration (P) prediction: (a) training time (seconds) *T*; (b) generalization performance index *J*.



(a)



(b)

Fig.7 Performance of the four IT2 FLS/FNN training algorithms on biochemical datasets for thalli concentration (X) prediction: (a) training time (seconds) *T*; (b) generalization performance index *J*.

REFERENCES

[1] L.X. Wang, *Adaptive fuzzy systems and control: design and stability analysis.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994

[2] J.S.R. Jang, C.T. Sun, and E. Mizutani, *Neuro-fuzzy and soft-computing*. Upper Saddle River, NJ: Prentice-Hall, 1997.

[3] J. Leski, "TSK-fuzzy modeling based on ε-insensitive learning," *IEEE Trans. Fuzzy Systems*, vol. 13, no.2, pp: 181-193, 2005

[4] Q. Liang, and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Trans. on Fuzzy Systems*, vol. 8, pp: 535-550, 2000.

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

[6] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Computational Intelligence Magazine*, vol. 2, pp: 20-29, 2007.

[7] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Systems*, vol.12, no. 1, pp: 84-98, 2004.

[8] Q. L. Liang and J. M. Mendel, "Designing interval type-2 fuzzy logic systems using an SVD-QR method: Rule reduction," *Int. J. Intell. Syst.* vol.15, no. 10, pp: 939-957, 2000.

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

[10] G. M. Mendez, "Orthogonal-back propagation hybrid learning algorithm for type-2 fuzzy logic systems," *IEEE Proceedings of the NAFIPS 04 International Conference on Fuzzy Sets*, 2: 899-902, 2004.

[11] C. H. 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. on Fuzzy Systems*, vol. 18, no. 4, pp:686-699, 2010.

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

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

[14] I. W. Tsang, J. T. Kwok and P. M. Cheung, "Core vector machines: fast SVM training on large data sets," *Journal of Machine Learning Research*, 6: 363-392, 2005.

[15] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Large-scale maximum margin discriminant analysis using core vector machines," *IEEE Trans. on Neural Net*works, vol. 19, no. 4, pp: 610-624, 2008.

[16] Z. H. Deng, F. L. Chung and S. T. Wang, "FRSDE: fast reduced set density estimator using minimal enclosing ball approximation," *Pattern Recognition*, vol.41, no.4, pp: 1363-1372, 2008.

[17] F. L. Chung, Z. H. Deng, and S. T. Wang, "From minimum enclosing ball to fast fuzzy inference system training on large datasets," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 1, pp:173-184, 2009.

[18] Z. H. Deng, K. S.Choi, F. L. Chung, S.T. Wang, "Scalable TSK fuzzy modeling for very large datasets using minimal enclosing ball approximation," *IEEE. Trans. Fuzzy systems*, vol.19, no.2, pp: 210-226, 2011,

[19] G. B. Huang, Zhu Q. Y., and C. K. SiewK, "Extreme learning machine:Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[20] G. B. Huang, L. Chen and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, 2006.

[21] H. J. Rong, G. B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1067–1072, Aug. 2009.

[22] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.

[23] G. Feng, G. B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, 2009.

[24] G. B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1–3, pp. 155–163, 2010.

[25] G. B. Huang, H. M. Zhou, X. J. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513-529, 2012.

[26] G. B. Huang, L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp: 3056–3062, 2007.

[27] G. B. Huang, L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol.71, pp:3460–3468, 2008.

[28] H. J. Rong, G. B. Huang, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. on Syst., Man, and Cybern., Part B*, vol. 39, no. 4, pp: 1067-1072, 2009.

[29] J. W. Cao, Z. P. Lin, G. B. Huang, "Composite function wavelet neural networks with differential evolution and extreme learning machine," *Neural Processing Letters*, vol. 33, no. 3, pp: 251-265, 2011.

[30] J, W. Cao, Z. P. Lin, G. B. Huang, "Composite function wavelet neural networks with extreme learning machine," *Neurocomputing*, vol. 73(7-9), pp:1405-1416, 2010.

[31] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Trans Inf. Theory*, vol. 44, no. 2, pp:525–536, 1998.

[32] J. M. BenÃtez, A. Blanco, M. Delgado, and I. Requena, "Neural methods for obtaining fuzzy rules," *Mathw. Soft Comput.*, vol. 3, no. 3, pp. 371–382, 1996.

[33] S. Horikawa, T. Furuhashi, S. Okuma, and Y. Uchikawa, "A fuzzy controller using a neural network and its capability to learn expert's control rules," *in Proc. IIZUKA*, 1990, pp. 103–106.

[34] J. M. Benitez, J. L. Castro, and I. Requena, "Are artificial neural networks black boxes," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1156–1164, Sep. 1997.

[35] F. L. Chung, S. T. Wang, Z. H. Deng, and D. W. Hu, "CATSMLP: towards a tobust and interpretable multilayer perceptron with sigmoid activation functions," *IEEE Trans. on Syst., Man and Cybern., Part B*, vol.36, no.6, pp.1319-1331, December 2006.

[36] D. Serre, *Matrices: Theory and Applications*. New York: Springer-Verlag, 2002.

[37] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1971.

[38] G. B. Huang, M. B. Li, L. Chen, C. K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, pp. 576–583, 2008.

[39] M. B. Li, G. B. Huang, P. Saratchandran, N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, pp. 306–314, 2005.

[40] G. B. Huang, P. Saratchandran, N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans Syst., Man and Cybern., Part B*, vol. 34, no.6, pp. 2284-2292, 2004.

[41] N. Y. Liang, G. B. Huang, P. Saratchandran, N. Sundararajan, "A fast and accurate on-line sequential learning algorithm for feedforward networks," *IEEE Trans Neural Netw*, vol. 17, no.6, pp. 1411–1423, 2006.

[42] M. van Heeswijk, Y. Miche, E. Oja A. Lendasse, "Gpu accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, no. 16, pp. 2430-2437, 2011.

[43] Y. Sun, Y. Yuan, G. Wang, "An OS-ELM based distributed ensemble classification framework in p2p networks," *Neurocomputing*, vol. 74, no. 16, pp. 2438-2443, 2011.

[44] Y. Lan, Y. C. Soh, G. B Huang, Ensemble of online sequential extreme learning machine. *Neurocomputing*, vol. 72, pp: 3391–3395, 2009.

[45] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *Ann. Stat.*, vol. 19, no. 1, pp. 1–141, 1991.

[46] L. Torgo. (2009). "Regression datasets," Dep. Comput. Sci., Porto Univ., Porto, Portugal. [Online]. Available: http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html.

[47] C. C. Chang and C. J. Lin, "LIBSVM : a library for support vector machines," *ACM Trans. on Intel. Syst. and Tech.*, vol. 2, pp. 27:1--27:27, 2011.

[48] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Inf. Sci*., vol. 132, pp. 195–220, 2001.

[49] D. Wu and J. M. Mendel, "Enhanced Karnik-Mendel Algorithms," *IEEE Trans. Fuzzy Systems*, vol. 17, pp. 923-934, 2009.

[50] D. Wu and M. Nie, "Comparison and Practical Implementation of Type-Reduction Algorithms for Type-2 Fuzzy Sets and Systems," *IEEE International Conference on Fuzzy Systems*, Taipei, Taiwan, June 2011.

[51] G. B. Huang, H. M. Zhou, X. J. Ding, R. Zhang, "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. on Syst., Man, and Cybern., Part B*, vol. 42(2), pp. 513-529, 2012.

[52] L. A. Zadeh, "Fuzzy sets", *Inf. Control*, vol. 8, pp. 338-353, 1965.

[53] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," *Commun. ACM*, vol. 37, pp. 77–84, 1994.

[54] J. R. Castro, O. Castillo, P. Melin, A. R. Díaz, "A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks," *Inf. Sci.*, vol. 179(13), pp. 2175-2193, 2009.

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

[56] O. Castillo, P. Melin, W. Pedrycz, "Design of interval type-2 fuzzy models through optimal granularity allocation," *Appl. Soft Comput.*, vol. 11(8), pp. 5590-5601, 2011.

[57] P. Melin, O. Mendoza, O. Castillo, "Face Recognition With an Improved Interval Type-2 Fuzzy Logic Sugeno Integral and Modular Neural Networks," *IEEE Trans. on Syst., Man, and Cybern., Part A*, vol. 41(5), pp. 1001-1012, 2011.

[58] P. Melin, O. Mendoza, O. Castillo, "An improved method for edge detection based on interval type-2 fuzzy logic," *Expert Syst. Appl*., vol. 37(12), pp. 8527-8535, 2010.

[59] Y. Maldonado, O. Castillo, 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.

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

[61] O. Castillo, P. Melin, A. A. Garza, O. Montiel, R. Sepúlveda, "Optimization of interval type-2 fuzzy logic controllers using evolutionary algorithms," *Soft Comput*., vol. 15, no. 6, pp. 1145-1160, 2011.

[62] R. A. Aliev, W. Pedrycz, B.G. Guirimov, R.R. Aliev, U. Ilhan, M. Babagil, S. Mammadli, "Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization," *Inf. Sci.*, vol. 181, no. 9, pp. 1591–1608, 2011.

[63] R. Martinez, A. Rodriguez, O. Castillo, L.T. Aguilar, "Type-2 fuzzy logic controllers optimization using genetic algorithms and particle swarm optimization," *Proc. the IEEE International Conference on Granular Computing*, GrC 2010, 2010, pp. 724–727.

[64] W. H. R. Jeng, C. Y. Yeh, S. J. Lee, "General type-2 fuzzy neural network with hybrid learning for function approximation," Proc. the IEEE Conference on Fuzzy Systems, Jeju, Korea, 2009, pp. 1534–1539.

[65] G. M. Méndez and M. A. Hernandez, "Hybrid learning mechanism for interval A2-C1 type-2 non-singleton type-2 Takagi-Sugeno-Kang fuzzy logic systems," *Information Sciences*, vol. 220, pp. 149-169, 2013

[66] G. M. Méndez, A. Hernandez, A. Cavazos, M. T. Mata, "Type-1 non-singleton type-2 Takagi-Sugeno-Kang fuzzy logic systems using the hybrid mechanism composed by a Kalman type filter and back propagation methods," *Lecture Notes in Artificial Intelligence*, HAIS 2010, pp. 429-437, Volume 6076, 23 June 2010.

[67] G. M. Méndez, M. De los Angeles Hernandez, "Hybrid learning for interval type-2 fuzzy logic systems based on orthogonal least-squares and back-propagation methods," *Information Sciences*, vol. 179, no. 13, pp. 2146-2157, 2009.

[68] G. M. Méndez, M. De los Angeles Hernandez, "Interval type-2 non-singleton type-2 Takagi-Sugeno-Kang fuzzy logic systems using the hybrid learning mechanism recursive-least-square and back-propagation methods," In the 11th International Conference on Control Automation Robotics & Vision (ICARCV), pp. 710-714, 2010.

[69] Y. Yang, Y. Wang, X. Yuan, "Bidirectional extreme learn*ing machine for regression problem and its learning effectiveness," IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1498-1505, 2012.

[70] R. Zhang ; Y. Lan ; G. B. Huang ; Z. B. Xu, "Universal approximation of extreme learning machine with adaptive growth of hidden nodes," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no.2, pp. 365-371, 2012.

[71] Z. H. You, Y. K. Lei, L. Zhu, J. Xia, and B. Wang, "Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis," *BMC Bioinformatics*, vol. 14, no. 8, pp. 1-11, 2013.

[72] T. Liu, L. Hu, C. Ma, Z. Y. Wang, and H. L. Chen, H. L, "A fast approach for detection of erythemato-squamous diseases based on extreme learning machine with maximum relevance minimum redundancy feature selection," *International Journal of Systems Science*, (ahead-of-print), pp. 1-13, 2013.