An Adaptive Neuro-Fuzzy System With Integrated Feature Selection and Rule Extraction for High-Dimensional Classification Problems

Guangdong Xue, Qin Chang, Jian Wang, Senior Member, IEEE, Kai Zhang, Fellow, IEEE, and Nikhil Ranjan Pal, Fellow, IEEE

Abstract—A major limitation of fuzzy or neuro-fuzzy systems is their failure to deal with high-dimensional datasets. This happens primarily due to the use of T-norm, particularly, product or minimum (or a softer version of it). Thus, there are hardly any work dealing with datasets having features more than hundred or so. Here, we propose a neuro-fuzzy framework that can handle datasets with even more than 7000 features! In this context, we propose an adaptive softmin (Ada-softmin) which effectively overcomes the drawbacks of “numeric underflow” and “fake minimum” that arise for existing fuzzy systems while dealing with high-dimensional problems. We call it an adaptive Takagi–Sugeno–Kang (AdaTSK) fuzzy system. We then equip the AdaTSK system to perform feature selection and rule extraction in an integrated manner. In this context, a novel gate function is introduced and embedded only in the consequent parts, which can determine the useful features and rules, in two successive phases of learning. Unlike conventional fuzzy rule bases, we design an enhanced fuzzy rule base, which maintains adequate rules but does not grow the number of rules exponentially with features that typically happens for fuzzy neural networks. The integrated feature selection and rule extraction AdaTSK (FSRE-AdaTSK) system consists of three sequential phases: 1) feature selection; 2) rule extraction; and 3) fine-tuning. The effectiveness of the FSRE-AdaTSK is demonstrated on 19 datasets of which five are in more than 2000 dimension including two with more than 7000 features. This may be the first attempt to develop fuzzy rule-based classifiers that can directly deal with more than 7000 features without requiring separate selection of features or any other dimensionality reduction method.

Index Terms—Feature selection, gate function, high-dimensional classification, rule extraction, Takagi–Sugeno–Kang (TSK) fuzzy system.

I. INTRODUCTION

FUZZY systems have been successfully applied in many areas, such as control engineering [1], [2], [3], [4] and pattern recognition [5], [6], [7], [8], [9], [10], Takagi–Sugeno–Kang (TSK) fuzzy system [2], [3], one of two classical fuzzy systems (the other one is Mamdani-Assilian (MA) fuzzy system [1]), has powerful ability to model nonlinear problems. To overcome the difficulties in identifying the fuzzy rules, various neural networks have been proposed based on their promising learning performance [4], [5], [7], [9], [10], [11]. As a consequence, back-propagation algorithm [12] has become a popular scheme to optimize TSK fuzzy systems.

The computation of the firing strength of fuzzy rules is a necessary and important operation in fuzzy reasoning. The firing strength can be computed using any T-norm [13]. In [14], two kinds of T-norms have been used which include minimum [2] and product [7], [9], [10], [15]. The product T-norm is differentiable and frequently used for solving low-dimensional problems. Since minimum is not differentiable, it is rarely considered when using gradient-based algorithm for learning parameters of fuzzy systems. To remedy this drawback, different softer versions of minimum called softmin have been employed [5], [8], [16]; although, strictly speaking they are not T-norms. The softmin is differentiable and can be a good approximator to minimum. In [16], the authors adopted the product and softmin to compute the firing strength, separately. The simulation results revealed that these two T-norm-based systems performed similarly on four regression datasets. For high-dimensional problem, one can easily deduce that the firing strength computed by the product T-norm would reduce almost to zero even when each antecedent component has a high membership value. Sometimes, this phenomenon is called “numeric underflow” [10], which means that the firing strength is too small to be correctly represented and used by the machine.

Some previous works using fuzzy model to deal with high-dimensional datasets have been done. Lee [17] used a rough set-based method to reduce the number of input features before using them to build the adaptive-network-based fuzzy inference system architecture. This alleviates the problem of “numerical underflow” to some extent, but still can not handle very high-dimensional data. In [18], the authors proposed a coevolutionary approach to generate sets of MA fuzzy systems for handling

Manuscript received 6 April 2022; revised 26 August 2022; accepted 3 November 2022. Date of publication 10 November 2022; date of current version 30 June 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFA0708700, in part by the National Natural Science Foundation of China under Grant 62173345, in part by the Major Scientific and Technological Projects of China National Petroleum Corporation (CNPC) under Grant ZD2019-183-008, and in part by the Fundamental Research Funds for the Central Universities under Grant 20CX05002A and Grant 22CX03002A. (Corresponding author: Jian Wang.) Guangdong Xue is with the College of Control Science and Engineering, China University of Petroleum (East China), Qingdao 266580, China (e-mail: g.d.xue@foxmail.com).

Qin Chang and Jian Wang are with the College of Science, China University of Petroleum, Qingdao 266580, China (e-mail: chq_163.com; wangjian@upc.edu.cn).

Kai Zhang is with the College of Petroleum Engineering, China University of Petroleum, Qingdao 266580, China (e-mail: zhangkai@upc.edu.cn).

Nikhil Ranjan Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta, West Bengal 700108, India (e-mail: nrpal59@gmail.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TFUZZ.2022.3220950.

Digital Object Identifier 10.1109/TFUZZ.2022.3220950
high-dimensional and large datasets. However, this work truly does not deal with high-dimensional data. In fact the number of input features of the two datasets used in this work are only 18 and 40, which are much less than what we mean by high-dimensional data (more than 1000 features). Moreover, triangular fuzzy sets and the product T-norm are used in this approach, so the aforementioned “numeric underflow” problem, even for moderately high-dimensional datasets, cannot be avoided. Typically, in a TSK model, the system output is computed as a convex combination of outputs of all rules, where each rule output is weighted by the normalized firing strength. To avoid the problem of saturation in case of high-dimensional data, authors in [19] computed the firing strength as an inverse function of the logarithm of the T-norm of the degree of satisfaction of the atomic antecedent clauses in the rule. This helps to avoid the underflow problem for high-dimensional data. But the use of such a formula destroys the interpretation of the AND operator and the semantics of “firing strength” of a rule is also lost. Moreover, the firing strength could even be more than one (1.0). In [20], authors discussed various ways to avoid the underflow problem. The authors propose to enhance the scale of the spread, σ, of the Gaussian membership function, where σ is computed using the fuzzy c-means outputs. The scaling is done by \( \sqrt{D} \), where D is the dimension of the data. This is equivalent to making the effective spread of the membership function \( \sqrt{D}\sigma \). Thus, for high-dimensional data, the spread will be large enabling the system to avoid the underflow problem. Of course, it reduces the specificity of the rules to some extent.

On the other hand, the softmax may also suffer from the same obstacle, “numeric underflow.” It also has another problem, which may be called as “fake minimum” that arises when the parameter of the softmax is not properly set. Thus, it is very important to design a novel adaptive version of softmax which can avoid these shortcomings to deal with high-dimensional problems.

For solving high-dimensional classification problems, feature selection becomes a necessity because when the input dimension is high, it is very likely that there are some redundant and derogatory features, which are expected to negatively affect the final performance [21]. In addition to improving the performance, feature selection can also simplify the complexity, lower the computational cost, and enhance the interpretability of the system. Generally, feature selection methods are classified into three groups: filter methods [22], wrapper methods [23], [24], [25], and embedded methods [6], [7]. A filter method selects features according to an evaluation criteria such as mutual information or correlation which is independent of the classifier or predictor, that will finally use these features. Although, filter methods are efficient in computation, they may produce unsatisfactory results because they ignore the feedback of the model [26] as well as the interaction between features. A wrapper method considers different feature subsets and then evaluates the goodness of them using classification or prediction accuracy. This results in good performance but costs more computational resource [27]. Moreover, to get the best subset, we need to evaluate all possible subsets of features, which is not feasible for high-dimensional datasets. An embedded method [6] simultaneously performs the feature selection and evaluation of the model and consequently, it enhances the effectiveness and reduces the computational burden. Additionally, some hybrid methods [28], [29] which utilize two of these three feature selection approaches together have also been investigated. In [28], the partial mutual information-based filter approach was coupled with a firefly algorithm-based wrapper method to deal with short-term load forecasting problem. And in [29], a hybrid wrapper-embedded feature selection method was proposed combining genetic algorithm and the embedded regularization approach.

Basically, there are four kinds of features as follows [30]:
1) essential features;
2) redundant features;
3) derogatory or bad features;
4) indifferent features.

Essential features are necessary for solving the problem, while derogatory or bad features have adverse effects. Indifferent features have neither positive nor negative influence on the tasks, and as for redundant features, they are useful but not all of them are needed. Feature selection method should select essential features, discard bad and indifferent features, and control the use of redundant features [31]. In [7] and [21], features are selected using a fuzzy rule-based framework and neural networks, respectively, and both of these works considered controlling the level of redundancy among the selected features. In [6] and [32], the authors did not take into consideration redundant features and got comparable results as well.

For mixed data, authors in [22] proposed an unsupervised spectral feature selection method, which computes a feature relevance score combining a kernel and a spectrum-based feature evaluation measure. Compared with other filter methods, the experimental results demonstrated promising performance. Kabir et al. [25] integrated a wrapper approach into neural networks which automatically determined the architecture of the network model during the feature selection process. In [21] and [32] embedded feature selection methods were proposed, which added the group lasso regularization to the loss function of neural networks to select useful features simultaneously with the training of the network. In [4], [5], [16], and [30], feature modulators were designed and used to modify the membership values in the antecedent parts of fuzzy rules for feature selection. Note that the feature modulator was introduced as a gate function in [16] which modified both antecedent and consequent parts, and finally picked up the required features. Actually, there are three commonly used gate functions to measure the magnitude of feature’s importance, \( 1 - e^{-\lambda^2} \) ([4], [5]), \( \frac{1}{1+e^{-\lambda^2}} \) and \( e^{-\lambda^2} \) ([6], [7], [31]), where \( \lambda \) is the tuneable gate parameter. In the existing works, the gate values (the values of gate function) are generally initialized around zeros, and the corresponding derivatives are close to zeros as well. This leads to more training iterations to open the gates during feature selection. Thus, designing of a suitable gate function to overcome this drawback is an important task.

For extraction of fuzzy rules, a variety of approaches have been proposed, such as heuristic approaches [33], [34], genetic algorithms-based methods [35], [36], [37], and neuro-fuzzy techniques [5], [16]. In [33], an efficient method was proposed to extract fuzzy rules directly from input data through activation and inhibition hyperboxes. Ishibuchi et al. [34] used a distributed representation of fuzzy rules for classification of patterns. This approach used multiple fuzzy partitions simultaneously to generate rules for classification. In [36], Ishibuchi and Yamamoto proposed a two-phase method to select a small number of simple rules. In phase I, the candidate rules were generated using two
metrics (confidence and support) and in phase II, the appropriate fuzzy rules were selected by a multiobjective genetic local search algorithm. In a neuro-fuzzy framework [5], all possible fuzzy rules were initially employed but after the training only an adequate set of rules involving a set of selected features was retained.

Depending on the method of extraction, fuzzy rule bases can be divided into two categories, compactly combined fuzzy rule base (CoCo-FRB) and fully combined fuzzy rule base (FuCo-FRB). After the feature space is clustered or partitioned, CoCo-FRB converts each cluster into one fuzzy rule; in other words, the number of rules is equal to the number of clusters. While FuCo-FRB considers all possible rules. Here on the domain of each feature, a number of linguistic values are defined and all possible valid combinations that can define rules are considered. Hence, the number of rules in FuCo-FRB exponentially increases with the number of input features. Thus, for high-dimensional problems, use of FuCo-FRB becomes prohibitive. In [9], a promising scheme was proposed to effectively address this problem. It first used principle component analysis to reduce the original number of input features to 5, and then the rule base was initialized to FuCo-FRB. Inspired by DropOut [38] and DropConnect [39], the so-called DropRule strategy was adopted in each iteration which randomly discarded the rules based on the drop rate. The simulation results demonstrated the superiority of this method over its counterparts. However, this scheme inevitably affects the interpretability of the constructed system.

In this article, we focus on TSK neuro-fuzzy systems to deal with high-dimensional classification problems. We introduce an adjustable-parameter-based softmin to compute the firing strength. Additionally, we propose a gate function to realize feature selection and rule extraction. As a result, we obtain a comprehensive feature selection and rule extraction-based system, that we call FSRE-AdaTSK. FSRE-AdaTSK consists of three sequential phases: 1) feature selection; 2) rule extraction; and 3) fine tuning. The main contributions in this work are summarized as follows.

1) An adaptive softmin called as Ada-softmin is proposed whose index parameter is adjusted on the basis of current membership values. Ada-softmin is able to approximate the minimum T-norm better than traditional softmin and it is adopted to compute the firing strength of rules for the TSK fuzzy system which results in the so called adaptive TSK (AdaTSK) model. The AdaTSK model effectively avoids two typical problems, “numeric underflow” and “fake minimum,” and thereby equips AdaTSK with the ability of dealing with high-dimensional data.

2) Inspired by the existing gate functions [4], [5], [6], [7], [31], a novel gate function is designed. When the gate values are initialized around zeros, the derivatives of the proposed gate function are much greater than those of the existing ones. This enables the proposed AdaTSK system to accomplish the feature selection task more efficiently than the traditional gate functions.

3) Based on CoCo-FRB, we develop an enhanced fuzzy rule base (En-FRB). It collects more fuzzy rules than CoCo-FRB and avoids the exponential growth in the number of rules caused by FuCo-FRB. Finally, for the constructed En-FRB, we propose an efficient embedded rule extraction method using the proposed gate function.

4) Using AdaTSK, the novel gate function, and En-FRB, we propose our final approach, i.e., FSRE-AdaTSK, which included feature selection and rule extraction procedures.

5) The proposed gate function and the adaptive softmin can be easily adapted to design fuzzy rule-based systems using both MA model and TSK model.

The rest of this article is organized as follows. CoCo-FRB-based TSK fuzzy system is reviewed and the AdaTSK method is introduced in the next section. Section III elaborates the details of the proposed FSRE-AdaTSK system. Section IV demonstrates the effectiveness of FSRE-AdaTSK system. Finally, Section V concludes the article and possible future research directions are also discussed.

II. TSK Fuzzy system With AdaTSK

In this section, we will first review the CoCo-FRB-based TSK fuzzy system for classification problems, and then introduce an adaptive softmin to construct the AdatSK system. For convenience, the notations used in this paper are listed in Table I.

A. CoCo-FRB-Based TSK Fuzzy System

Consider a classification problem involving C classes. Let an instance or data point be represented by a D-dimensional feature vector, i.e., \( x = (x_1, x_2, \ldots, x_D)^T \in \mathbb{R}^D \). We assume that each feature is associated with S fuzzy sets. For a CoCo-FRB-based TSK fuzzy system, the number of fuzzy rules, \( R \), is equal to \( S^C \). Generally, the \( r \)th (\( r = 1, 2, \ldots, R \)) fuzzy rule of the first-order TSK model for classification problems is described as:

\[
\begin{align*}
\mu_{r,d}(x) &= \text{membership value of the } d\text{th feature of } x \text{ in the } r\text{th rule} \\
f_r(x), \bar{f}_r(x) &= \text{firing strength and normalized firing strength of the } r\text{th rule for } x \\
y_r^c(x) &= \text{output of the } r\text{th rule of the } c\text{th class for } x \\
y_c(x) &= \text{cth component of true label of } x \text{ coded by one-hot coding} \\
E &= \text{system error} \\
N &= \text{number of training instances} \\
\hat{q} &= \text{parameter of Ada-softmin} \\
\eta &= \text{learning rate} \\
I &= \text{index matrix to calculate firing strength} \\
\lambda, \theta &= \text{parameters of gate function for feature selection and rule extraction} \\
\tau &= \theta \text{threshold of feature selection and rule extraction} \\
\zeta &= \text{coefficient to compute threshold}
\end{align*}
\]
as below

\[
\text{Rule}_r : \text{IF } x_1 \text{ is } A_{r,1} \text{ and } \cdots \text{ and } x_D \text{ is } A_{r,D},
\]

THEN \( y_r^1(x) = p_{r,0}^1 + \sum_{d=1}^{D} p_{r,d}^1 x_d \), \( \cdots \)

\[
y_r^C(x) = p_{r,0}^C + \sum_{d=1}^{D} p_{r,d}^C x_d
\]

(1)

where \( A_{r,d} \) is the \( r \)th fuzzy set associated with the \( d \)th feature used in the \( r \)th rule, \( y_r^c(x) \) \((c = 1, 2, \ldots, C)\) is the output of the \( r \)th rule for the \( c \)th class computed from \( x \), and \( p_{r,d}^c \) represents a consequent parameter of the \( r \)th rule associated with the \( d \)th feature for the \( c \)th class.

There are plenty of choices of membership functions (MFs) that can be used for \( A_{r,d} \), such as triangular, trapezoidal, and Gaussian functions [6], [7]. We note that the Gaussian MF is frequently employed due to its smooth and infinitely differentiable properties. Specifically, the membership value of \( x_d \) on \( A_{r,d} \) is

\[
\mu_{r,d}(x) = \exp \left( \frac{-(x_d - m_{r,d})^2}{2\sigma_{r,d}^2} \right)
\]

(2)

where \( m_{r,d} \) and \( \sigma_{r,d} \) represent, respectively, the center and spread of the \( r \)th Gaussian membership function defined on the \( d \)th input variable. Note that, although in (2) the argument of \( \mu_{r,d}(x) \) is shown as \( x \), the function only uses the \( d \)th component of \( x \). In [40], the authors used \( \mu = \exp \left( \frac{-|x|^2}{\sigma^2} \right) \) as membership function but set \( \sigma = 1 \) for all fuzzy subsystems which means (2) is simplified as

\[
\mu_{r,d}(x) = \exp \left( -(x_d - m_{r,d})^2 \right).
\]

(3)

Following [40], the membership values are evaluated by (3) in this article. Clearly, \( 0 < \mu_{r,d}(x) \leq 1 \).

Since minimum is not differentiable, the product is often used to compute the firing strength [4], [6], [7], [9], [10], [15], [41].

\[
f_r(x) = \sum_{d=1}^{D} \mu_{r,d}(x)
\]

(4)

where \( f_r(x) \) is the firing strength of the \( r \)th rule for the input \( x \). Note that the product T-norm often leads to a serious problem when the number of features, \( D \), is large. It easily makes \( f_r(x) \) very close to 0 even beyond the scope that machines can identify, i.e., “numeric underflow” [10].

The output of the \( r \)th rule associated the \( c \)th class computed from \( x \) is

\[
y_r^c(x) = p_{r,0}^c + \sum_{d=1}^{D} p_{r,d}^c x_d.
\]

(5)

If the normalized firing strength is defined as

\[
\bar{f}_r(x) = \frac{f_r(x)}{\sum_{r=1}^{R} f_r(x)}
\]

then, the \( c \)th component of the system output on \( x \) is

\[
y^c(x) = \frac{R}{\sum_{r=1}^{R} \bar{f}_r(x)} y_r^c(x).
\]

(7)

For the sake of clarity, the neural network structure implementing CoCo-FRB-based TSK fuzzy system described above is shown in Fig. 1. By contrast, FuCo-FRB considers all possible valid combinations of fuzzy sets defined on each feature, in which the number of rules exponentially increases with features. These two fuzzy rule bases will be illustrated and compared in Section III-B.

Assume that there are \( N \) training samples. There are several choices of error functions [6], [9], [10] that can be used to update system parameters based on the back-propagation of error or gradient descent (GD) algorithm [12], [42], [43]. The typical mean square error function is used in this article

\[
E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{c=1}^{C} (y_c^r(x_n) - y_c(x_n))^2
\]

(8)

where \( y_c^r(x_n) \) and \( y_c(x_n) \), respectively, correspond to the \( c \)th component of the system output and the true label vector for \( x_n \) \((n = 1, 2, \ldots, N)\). The gradients of the error function (8) with respect to the antecedent and consequent parameters are derived as

\[
\frac{\partial E}{\partial m_{r,d}} = \frac{1}{N} \sum_{n=1}^{N} \left[ 2f_r(x_n)(x_{n,d} - m_{r,d}) \right.
\]

\[
\times \left. \sum_{c=1}^{C} \left( (y_c^r(x_n) - y_c(x_n)) (y_c^r(x_n) - y_c^c(x_n)) \right) \right],
\]

(9)

\[
\frac{\partial E}{\partial p_{r,d}^c} = \frac{1}{N} \sum_{n=1}^{N} \left[ (y_c^r(x_n) - y_c(x_n)) f_r(x_n) x_{n,d} \right],
\]

(10)

where \( x_{n,d} \) is the \( d \)th component of \( x_n \). Then, the update formula of system parameters in the \( k \)th iteration is as follows:

\[
\omega(k+1) = \omega(k) - \eta \sum_{c=1}^{C} \frac{\partial E}{\partial \omega_c(k)}
\]

(11)

where \( \omega \) indicates the general parameters of antecedent and consequent parts, \( \eta > 0 \) is the learning rate.

\section{Ada-Softmin}

It is easy to observe that the firing strength evaluated by the product T-norm takes a very small positive value even for problems with a moderate number of features. Thus, when the number of input features is considerably large, the computation of firing strength using product may encounter underflow. To deal with this issue and also to overcome the nondifferentiability of minimum, the following softmin as a substitute of minimum is often used to compute the firing strength [5], [16]:

\[
\text{softmin}(v_1, v_2, \ldots, v_D, q) = \left( \frac{v_1^q + v_2^q + \cdots + v_D^q}{D} \right)^{\frac{1}{q}}.
\]

(12)

As \( q \rightarrow -\infty \), the softmin tends to the minimum of \( v_i \) \((i = 1, 2, \ldots, D)\). Practically, the parameter \( q \) is often set to be a fixed constant such as −12 in [5] and −11 in [16]. Unfortunately, the softmin with fixed parameter has two drawbacks: 1) numeric underflow; and 2) fake minimum. Similar to [5], we set \( q = -12 \) and use the following two examples to demonstrate the limitations of softmin.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 1. CoCo-FRB-based TSK fuzzy system implemented by neural networks. The first layer is input layer, of which the outputs are the values of the input features. In the second layer, \( A_i \ (i = 1, 2, \ldots, S; \ j = 1, 2, \ldots, D) \) is the \( j \)-th fuzzy set defined on the \( i \)-th feature, and the output of each node is computed by (3). For each node in the third layer, \( \mathcal{D} \) membership values are used together to compute a firing strength by a T-norm. The firing strengths are normalized though \( \hat{\mathcal{D}} \) is actually shown as \( \mathcal{D} \). The firing strength is calculated with suitable negative integer, membership values, the parameter, the simulation result is with overflow" since the fact that \( \mathcal{D} \) lies as the lower bound of \( \mathcal{D} \), is not greater \( (6) \) in the fourth layer. The two fully connected layers in the lower part represent the consequent parts, i.e., the “THEN” parts described in (1). The defuzzification process, shown in (7), is denoted by the last two layers.

Numeric underflow: Assume that

\[
u_1 = 1.1 \times 10^{-26}, \quad u_2 = 1.8 \times 10^{-22}, \quad u_3 = 1.5 \times 10^{-9}.
\]

According to (12), theoretically, \( u_{12}^{-1} \) should be equal to 0.3186 \( \times 10^{312} \). However, this is significantly beyond the computation scope of the machine (We are using a 64-bit processor). The evaluation of \( u_{12}^{-1} \) is actually shown as Inf\(^1\) during running the program code. This then leads to the phenomenon “numeric underflow,” i.e.,

\[
\text{softmin}(u_1, u_2, u_3, -12) = 0.
\]

Fake minimum: Assume that

\[
v_1 = 0.5, \quad v_2 = 0.55, \quad v_3 = 0.49, \quad v_4 = 0.48
\]

the simulation result is with

\[
\text{softmin}(v_1, v_2, v_3, v_4, -12) = 0.49777.
\]

It is easy to see that softmin fails to get the minimum, 0.48, and computes a value even bigger than the second minimum, \( v_3 = 0.49 \).

Now, we introduce an adaptive softmin called as Ada-softmin to effectively solve the above two problems. Based on the current membership values, the parameter, \( q \), is adaptively assigned a suitable negative integer, \( \hat{q} \), instead of a fixed value. Specifically, the firing strength is calculated with

\[
f_r(x) = \left( \frac{\mu_{r,1}(x) + \mu_{r,2}(x) + \cdots + \mu_{r,D}(x)}{D} \right)^{\frac{1}{\hat{q}}} \tag{15}
\]

where

\[
\hat{q} = \left[ \frac{690}{\ln(\min\{\mu_{r,1}(x), \mu_{r,2}(x), \ldots, \mu_{r,D}(x)\})} \right] \tag{16}
\]

\(^1\) From this point of view, the described issue can also be called “numeric overflow” since the fact that \( u_1^{-1} \) is 0.3186 \( \times 10^{312} \). and \( \lceil \cdot \rceil \) is the ceiling function. We note that the adaptive parameter, \( \hat{q} \), is deduced by the assumption: each \( \mu_{r,d}(x) \) is not greater than 10\(^{300}\). This clearly guarantees that the value of \( \mu_{r,d}^\hat{q}(x) \) lies in the scope of a 64-bit machine. By the proposed Ada-softmin, one can easily obtain the new results on (13) and (14)

\[
\begin{align*}
\hat{q}_2 &= \left[ \frac{690}{\ln(\text{min}\{u_1, u_2, u_3\})} \right] = -11, \\
\text{softmin}(u_1, u_2, u_3, \hat{q}_2) &= 1.2 \times 10^{-26} \\
\hat{q}_1 &= \left[ \frac{690}{\ln(\text{min}\{v_1, v_2, v_3, v_4\})} \right] = -940, \\
\text{softmin}(v_1, v_2, v_3, v_4, \hat{q}_1) &= 0.4807.
\end{align*}
\]

As shown in (17) and (18), the proposed Ada-softmin can get better approximations to the minimum in these two cases. For our experiments, we set -1000 as the lower bound of \( q \). Specifically, if the \( q \) calculated by (16) is less than -1000, we set \( q \) to -1000. Although theoretically, one does not need to set this lower bound, we use it to avoid underflow in practice. Since the proposed Ada-softmin has the ability to adaptively find the appropriate parameter of the softmin, the AdaTSK can be successfully used for the high-dimensional problems. The classification results of TSK fuzzy systems with product, softmin, and Ada-softmin are compared in Section IV-B, which demonstrate the effectiveness of AdaTSK.

III. FSRE-AdaTSK

In this section, we introduce a new gate function which is embedded in AdaTSK fuzzy system for feature selection and rule extraction. For feature selection, the fuzzy rule base, CoCo-FRB, is considered. While for rule extraction, an enhanced fuzzy rule base called as En-FRB is designed inspired by CoCo-FRB and FuCo-FRB. To some extent, En-FRB can be regarded as a compromised solution between them.
A. Gate Function

Each T-norm, $T$, satisfies the equation $T(1, \mu) = \mu$, which means 1 has no effect on the result of T-norms. In a fuzzy rule-based system, if the membership value of a feature always becomes 1 irrespective of the actual numerical value of the feature, then this feature makes no difference to the firing strength. Some of the earlier studies used a gate function to modulate membership value of each feature in such way that the features whose modulated membership value is nearly 1 can be removed from the feature space and rule base. For example, assume that the membership value $\tilde{\mu}$ is calculated from the feature $F$ using (3). The gate function $M(\lambda)$ can modulate $\tilde{\mu}$ as $\tilde{\mu}^M(\lambda)$. We can then use GD-based algorithm to learn the gate parameter $\lambda$. When $M(\lambda) = 0$ the feature $F$ has no contribution to the firing strength and consequently is pruned. Every feature is equipped with a gate function and the features with high gate values are important features and are selected [4], [5], [6], [7], [16], and [30]. In addition, for the first-order TSK fuzzy model, the gate function needs to be embedded on both antecedent and consequent parts for feature selection [16].

Note: In this article we take a completely different philosophy. We do not modify the antecedent membership values using gate function as done in [4], [5], [6], [7], [16], and [30]. We use two families of gate functions. The first family ($M(\lambda)$) is used for feature selection, while the second family ($M(\theta)$) is used for rule extraction. These two families of gate functions are used in two different phases of the training and in both cases the gates are used only with the consequents.

In our proposed approach, following [5], [6], [31], at the beginning, all features are regarded as poor features, i.e., the gates, $M(\lambda)$s, of the features are initialized to low values. However, we note that for the commonly used gate functions, such as $M(\lambda) = \frac{1}{1 + e^{-s}}$, $M(\lambda) = 1 - e^{-\frac{s}{2}}$ and $M(\lambda) = e^{-\frac{s}{2}}$, their respective derivatives are close to zero, when the gate values are close to zero. This makes the learning very slow at the beginning. Motivated by this observation, we propose the following gate function:

$$M(\lambda) = \lambda \sqrt{e^{1-\lambda^2}} \quad (19)$$

where $\lambda$ is the gate parameter.

Fig. 2(a) pictorially depicts the plots of the four gate functions that we have discussed. Since initially all gates are assumed to be almost closed, the corresponding parameters, $\lambda$s, are located in different regions for different gate functions. The purple dashed-dotted line in Fig. 2(a) depicts $M(\lambda) = \frac{1}{1 + e^{-s}}$ [31] for which the $\lambda$s are initialized at $-5.0 \pm$ random noise in [0,1]. The plots of $M(\lambda) = 1 - e^{-\frac{s}{2}}$ [5] and $M(\lambda) = e^{-\frac{s}{2}}$ [6] are shown using the yellow dotted line and orange dashed line, respectively. For these two gate functions, the $\lambda$s are initialized with 0.001 and $3.0 + \mathcal{N}(0,0.2)$, respectively, where $\mathcal{N}(0,0.2)$ indicates the Gaussian noise with mean zero and standard deviation 0.2. The proposed gate function is depicted using the blue solid line in Fig. 2(a). For this function, the parameter values corresponding to a closed gate are located in three different zones: Around zero, around $-3.0$, and around $+3.0$. Besides, it is easy to see that this new gate function is an odd function. It is significantly different from the other three gate functions which lie always above the horizontal axis. As we can see from Fig. 2(a), $M(\lambda)$ could be negative. As far as the consequent is concerned, a high magnitude negative gate value will also indicate a useful feature. However, features with absolute gate values close to zero can be deleted. Thus, the proposed gate function can be viewed as a bidirectional door, one can open it by pushing it or pulling it in opposite direction. In other words, for (19), its absolute value can be denoted as the opening degree of the associated gate.

The derivatives of the above four gate functions are graphed in Fig. 2(b). For almost closed initial gates, the magnitudes of the derivatives of these four gate functions are shown using different symbols in Fig. 2(b), which are marked in the legend. Here, the parameter value of the proposed gate function is initialized around zero to realize an almost closed gate. As revealed by Fig. 2(b), the magnitude of the derivative at the five-pointed star is much higher than that at the remaining three symbols (two triangles and the circle), which implies that the magnitude of the derivative of the proposed gate function is much greater than those of its counterparts. Thus, this enables the proposed gate function to learn to distinguish between good and poor features faster compared to the other choices. To demonstrate the advantage of the proposed gate function, we choose the most representative gate function of the three existing ones, i.e., $M(\lambda) = e^{-\frac{s}{2}}$, to do a comparative study on the Wine dataset. The details are presented in Section IV-A. Moreover, an additional comparison with other gate functions is provided in Fig. S-1 and discussed in supplementary materials.
Fig. 5. Neural network structure of antecedent parts in CoCo-FRB and FuCo-FRB. (a) CoCo-FRB. (b) FuCo-FRB.

B. Enhanced Fuzzy Rule Base (En-FRB)

In this section, the enhanced fuzzy rule base, En-FRB, is constructed which is an intermediate case of CoCo-FRB and FuCo-FRB. The antecedent parts in CoCo-FRB and FuCo-FRB are shown with two different network architectures in Fig. 3. The number of rules in these two fuzzy rule bases are \( S \) and \( S^D \), respectively. FuCo-FRB considers all possible feasible rules in which all fuzzy sets corresponding to every feature are combined with each other, that is, \( R = S^D \). Clearly, FuCo-FRB is not a feasible approach for high-dimensional problems. While CoCo-FRB could be regarded as a simplified case of FuCo-FRB, \( R = S \). None of these two types of fuzzy rule bases may be suitable for solving practical problems because CoCo-FRB may not be expressive enough while FuCo-FRB is too big to consider. However, if we could choose the right number of clusters CoCo-FRB may solve the problem, but choosing the right number of clusters is a difficult problem.

To construct the En-FRB, we first introduce the following index matrix (I-matrix) to represent the combination of fuzzy sets. Note that I-matrix is made up of the first subscript of \( A_{i,j} (i = 1, 2, \ldots, S; j = 1, 2, \ldots, D) \), which represents the \( i \)th fuzzy set defined on the \( j \)th feature. For CoCo-FRB shown in Fig. 3(a), the I-matrix is as follows:

\[
I_{\text{CoCo}} = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
2 & 2 & \cdots & 2 \\
\vdots & \vdots & \ddots & \vdots \\
S & S & \cdots & S
\end{bmatrix}_{S \times D}
\]

where \( I_{\text{CoCo}}(i, j) (i = 1, 2, \ldots, S; j = 1, 2, \ldots, D) \) is the index of the fuzzy set associated with the \( j \)th feature in the \( i \)th rule. Each row of \( I_{\text{CoCo}} \) represents a fuzzy rule. For example, \( [22 \cdots 2] \), represents the antecedent of a rule involving the second fuzzy set defined on each of the \( D \) features. The I matrix, \( I_{\text{FuCo}} \) for

FuCo-FRB is obtained by extending \( I_{\text{CoCo}} \) as

\[
I_{\text{FuCo}} = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
S & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & S \\
S & S & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
S & S & \cdots & S
\end{bmatrix}_{S^D \times D}
\]

To avoid this exponential growth of the number of rules in \( I_{\text{FuCo}} \), we extend \( I_{\text{CoCo}} \) to generate \( I_{\text{En}} \) as follows. We begin with \( I_{\text{CoCo}} \) and change each element of \( I_{\text{CoCo}} \) by subtracting 1 or adding 1 for it to construct new fuzzy rules. For example, \( [2222] \) represents a fuzzy rule with four features. Subtracting 1 from each 2 one by one, we can obtain 4 rules, i.e., \([1222], [2122], [2212], \) and \([2221]\). On the other hand, adding 1 on each 2 one by one, we can get another 4 rules, \([3222],[2322],[2232] \) and \([2223]\). Each element in \( I_{\text{CoCo}} \) can be changed by subtracting 1 or adding 1 to construct new fuzzy rules, which will represent the input space near to the input spaces represented by \( I_{\text{CoCo}} \). In this way, we derive En-FRB. However, this may lead to some out of range values, which is handled as follows. If there are \( S \) linguistic values defined on each variable, then if we subtract 1, there will be some 0 values which will be replaced by \( S \). Similarly, if we add 1, some values will be \( S + 1 \) and those will be replaced by 1. Finally, we can obtain the I-matrix of En-FRB, which is

\[
I_{\text{En}} = \begin{bmatrix}
1 & 1 & \cdots & 1 \\
S & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & S \\
2 & 1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
1 & 1 & \cdots & 2 \\
S & S & \cdots & S \\
S - 1 & S & \cdots & S \\
\vdots & \vdots & \ddots & \vdots \\
S & S & \cdots & S - 1 \\
1 & S & \cdots & S \\
\vdots & \vdots & \ddots & \vdots \\
S & S & \cdots & 1
\end{bmatrix}_{(2D+1)S \times D}
\]

The En-FRB generates \( 2 \times D \) rules from each rule in CoCo-FRB and thus, \( 2DS + S = (2D + 1)S \) rules are obtained. As a consequence, the number of rules \( R \) in En-FRB increases linearly rather than exponentially with the number of features \( D \). Here, we point out that, for the special case \( D = 1 \) or \( S = 1 \), En-FRB degenerates into CoCo-FRB. On the other hand, if \( D = 2 \) or \( S = 2 \), the number of rules contained in En-FRB is...
(D + 1)S instead of (2D + 1)S since En-FRB includes part of repeated rules \((D \times S)\) rules.

Based on En-FRB, the rule extraction can be done using the TSK fuzzy system. For convenience, while writing the fuzzy rules, we use \(\bar{I}\) in place of \(I_{En}\) to indicate the I-matrix of En-FRB. Thus, the \(r\)th rule in En-FRB can be rewritten as follows:

\[
\text{Rule}_r: \quad \text{IF } x_1 = A_{I_{(r,1)}} \text{ and } \ldots \text{ and } x_D = A_{I_{(r,D)}}, \text{ THEN } \sum_{d=1}^{D} M(\theta_r)x_d
\]

where \(r = 1, 2, \ldots, R\) and \(R = (2D + 1)S\). The neural network structure of En-FRB-based TSK fuzzy system is shown in Fig. 4. The main difference between Figs. 1 and 4 is the connections between layers 2 and 3, the latter has more connections than the former between these two layers.

C. Feature Selection and Rule Extraction

We use the proposed gate function, \((19)\), for feature selection and rule extraction on the constructed AdaTSK system. For feature selection, the consequent parameters associated with the \(d\)th feature are multiplied by the corresponding gate function \(M(\lambda_d)\), thus, \((5)\) is modified as

\[
y_r^c(x) = p_{r,0}^c + \sum_{d=1}^{D} M(\lambda_d)p_{r,d}^c x_d
\]

where \(r = 1, 2, \ldots, R; c = 1, 2, \ldots, C\). On the other hand, for rule extraction, the entire consequent, i.e., all consequent parameters in the \(r\)th rule are multiplied by the rule-gate function \(M(\theta_r)\). Thus, \((5)\) is modified as

\[
y_r^c(x) = M(\theta_r)p_{r,0}^c + M(\theta_r) \sum_{d=1}^{D} p_{r,d}^c x_d
\]

where \(r = 1, 2, \ldots, R; c = 1, 2, \ldots, C\). \(\lambda_d\) and \(\theta_r\) are updated along with the system parameters. Note that these two families of gates are used in two sequential phases of training, not simultaneously.

As mentioned earlier, the final FSRE-AdaTSK scheme includes three sequential phases: 1) feature selection; 2) rule extraction; and 3) fine tuning. Specifically, in phase 1), we do the feature selection based on CoCo-FRB. And in phase 2), the En-FRB is constructed in the reduced feature space and the rule extraction is then implemented on it. Note that the rule extraction phase discards rules that are not useful. In phase 3), the system parameters are fine-tuned to improve the classification performance. The proposed gate function is used in both phases 1) and 2) and not in phase 3). The Ada-softmin is used in all of these three phases.

For feature selection, the gradient of error function with respect to the consequent parameters is rewritten as

\[
\frac{\partial E}{\partial p_{r,d}} = \frac{1}{N} \sum_{n=1}^{N} \left[ (y_r^c(x_n) - y_c(x_n)) \bar{f}_r(x_n) M(\lambda_d) x_{n,d} \right]
\]

On the other hand, for rule extraction, the gradient of error function with respect to the consequent parameters is derived as

\[
\frac{\partial E}{\partial p_{r,d}} = \frac{1}{N} \sum_{n=1}^{N} \left[ (y_r^c(x_n) - y_c(x_n)) \bar{f}_r(x_n) M(\theta_r) x_{n,d} \right]
\]
Algorithm 1: The FSRE-AdaTSK Algorithm.

**Input:** Training set along with class label of each instance; The number of fuzzy sets on each feature along with their definitions to define the initial rule base; The maximum number of training iterations in each of the three phases; The learning rate;

**Output:** The final system and gate parameters.

Step 1: Initialize CoCo-FRB and the parameters of the gate functions for feature selection. Calculate the forward propagation process by (3), (15), (6), (21), and (7), and compute system error by (8). Then, compute the gradients of the error with respect to the system parameters and gate parameters by (9), (23), and (26), and update the parameters a given number of iterations using (11).

Step 2: Compute the threshold $\tau_\eta$ by (28). Select the $d$th feature if $M(\lambda_d) > \tau_\eta$ and simplify the rule base.

Step 3: Reinitialize En-FRB using the selected features. Initialize the parameters of gate functions for rule extraction. Calculate the forward propagation process by (3), (15), (6), (22), and (7), and compute the system error by (8). Then, compute the gradients of the error with respect to the system parameters and gate parameters by (9), (24), and (27), and update the parameters a given number of iterations using (11).

Step 4: Compute the threshold $\tau_\theta$ by (29). Retain the rule, Rule $r$, if $\theta_r > \tau_\theta$. If the number of extracted rules is less than the number of classes, then retain $C$ rules with the highest gate values. The simplified fuzzy rule base is thus obtained.

Step 5: To refine the simplified fuzzy rule base, we compute the forward propagation process by (3), (5), (6), (7), and (15), compute system error by (8). Then, compute the gradients of the system error with respect to the centers and consequent parameters by (9) and (10), and update them a certain number of iterations using (11). This concludes design process of the rule base.

While in phase 3), since already the features and rules have been selected and the gates have been removed, the gradient of error function with respect to the consequent parameters is the same as in (10). Since the gate function has no effect on the antecedent part, the formula to compute the gradients of error function with respect to centers are the same in these three phases, which is presented in (25) shown at the bottom of this page. The (26) and (27) shown at the bottom of this page, show the gradients of the error function with respect to $\lambda$s and $\theta$s, respectively. All of these parameters are updated by the general update rule in (11).

We use a threshold, $\tau_\lambda$, to select useful features and another threshold, $\tau_\theta$, to extract important rules. The phase 1) training starts with almost closed gates for all features and the phase 2) training starts will almost closed gates for all rules. After training, the features and rules whose gate values are greater than their corresponding thresholds are selected in the two respective phases.

The thresholds for feature selection and rule extraction are computed as follows:

$$\tau_\lambda = \max_d \{M(\lambda_d)\} - \zeta_\lambda \left[ \max_d \{M(\lambda_d)\} - \min_d \{M(\lambda_d)\} \right],$$

$$\tau_\theta = \max_r \{M(\theta_r)\} - \zeta_\theta \left[ \max_r \{M(\theta_r)\} - \min_r \{M(\theta_r)\} \right]$$

where $\zeta_\lambda$ and $\zeta_\theta$ are the coefficients to compute thresholds for feature selection and rule extraction, respectively. For feature selection, if we use a bigger $\zeta_\lambda$, then more features will be selected/retained; while a low value of $\zeta_\lambda$ will remove more features. High-dimensional datasets generally are expected to have more redundant and not-useful features and, hence, demand removal of more features. Consequently, for high-dimensional data, we use a smaller $\zeta_\lambda$ to remove more undesirable features. We note here that this removal is done after the training, therefore, the modulators of the features that are required to solve the problem will attain the desired values and will not be eliminated.

For rule extraction, similarly, with bigger $\zeta_\theta$, more rules are selected. Generally, reduction of feature dimension makes the class boundaries more complex than those are expected in high dimension and for learning complex class boundaries usually more rules are required. Therefore, we adopt different thresholds for low and high-dimensional datasets, $\zeta_\lambda = 0.5$, $\zeta_\theta = 0.3$ for the low-dimensional datasets and $\zeta_\lambda = 0.4$, $\zeta_\theta = 0.5$ for high-dimensional datasets. Additionally, to select suitable thresholds, the cross-validation mechanism can be used. Since the primary objective of this study is not to generate a rule base achieving the best performance, we do not pursue this.

When the number of extracted rules is smaller than the number of classes, it may make the learning of the classifier more challenging. Thus, the lower bound on the number of the extracted rules is fixed as the number of classes, $C$. We note here that for the TSK classifier model, as we shall demonstrate

$$\frac{\partial E}{\partial m_{r,d}} = \frac{1}{N} \sum_{n=1}^{N_n} \left[ 2 \hat{f}_r(x_n)(x_{n,d} - m_{r,d}) \frac{\mu^\hat{\lambda}_{r,d}(x_n)}{\mu^\hat{\lambda}_{r,1}(x_n) + \cdots + \mu^\hat{\lambda}_{r,D}(x_n)} \sum_{c=1}^{C} \left[(y_c(x_n) - y_c(x_n))(y^\hat{c}_c(x_n) - y^\hat{c}_c(x_n))\right] \right]$$

$$\frac{\partial E}{\partial \lambda_d} = \frac{1}{N} \frac{\left(1 - \lambda^2_d\right)}{\sqrt{1 - \lambda_d^2}} \sum_{n=1}^{N_n} \sum_{r=1}^{R} \sum_{c=1}^{C} \left[ f_r(x_n)[y^\hat{c}_c(x_n) - y_c(x_n)]p^c_{r,d}x_{n,d} \right]$$

$$\frac{\partial E}{\partial \theta_r} = \frac{1}{N} \frac{\left(1 - \theta^2_r\right)}{\sqrt{1 - \theta^2_r}} \sum_{n=1}^{N_n} \sum_{c=1}^{C} \left[ f_r(x_n)[y^\hat{c}_c(x_n) - y_c(x_n)]p^c_{r,0} + \sum_{d=1}^{D} p^c_{r,d}x_{n,d} \right].$$
in Section IV-B, it is not necessary to have at least $C$ rules. We enforce this as it will facilitate the learning of the classifier.

IV. EXPERIMENTS AND RESULTS

To make these three sequential phases more clear, the steps of FSRE-AdaTSK are described in Algorithm 1. It clearly explains the formulae that are used in the different training phases of the proposed approach. To demonstrate the effectiveness of AdaTSK, 19 classification datasets are tested in the experiments. Table II summarizes the information on these datasets. Note that the last seven datasets with more than 1000 features are considered high-dimensional problems in this investigation.

For our experiments, we use the following computational protocols: For each feature we define $S$ (different values of $S$ are used in the following experiments) fuzzy sets, each modeled by a Gaussian membership function to have a simple fuzzy system. The centers of the membership functions are evenly placed on the domain of the feature as defined by the minimum and the maximum feature values in the training set. Let $V_m$ and $V_M$ be the minimum and maximum values of a feature, respectively. Once $S$ is given, the center of the $s$th membership function is obtained by $V_m + (V_M - V_m) \times \frac{s-1}{S-1}$, where $s = 1, 2, \ldots, S$. For example, assume $V_m$ and $V_M$ as $-1$ and $1$, respectively. If three fuzzy sets are defined on this feature, i.e., $S = 3$, then the centers are $\{-1, 0, 1\}$. All the consequent parameters are initialized to zero. We note here that after the first update cycle, the consequent parameters will take non-zero values. The same protocol is followed for all datasets. Then, we use full batch GD algorithm to optimize this AdaTSK system. We observed that for high-dimensional datasets, at the end of training, the centers of the membership functions do not change much from their initial values. The centers are barely updated significantly regardless of whether we use product, softmin, or Ada-softmin as the T-norm. Therefore, we keep the centers fixed at their initial values. The consequent parameters will take non-zero values. The same coefficient, it will exhibit a faster gate opening, which is an advantage.

| Dataset  | #Features | #Classes | Dataset Size |
|----------|-----------|----------|--------------|
| Iris     | 4         | 3        | 150          |
| Appendicitis | 7        | 2        | 106          |
| Pima     | 8         | 2        | 768          |
| Yeast    | 8         | 10       | 1484         |
| Glass    | 9         | 6        | 214          |
| Page-blocks | 10       | 5        | 5473         |
| Wine     | 13        | 3        | 178          |
| Heart    | 13        | 2        | 270          |
| Web     | 30        | 2        | 569          |
| Texture | 40        | 11       | 5500         |
| Spectrakeart | 44       | 2        | 267          |
| Sonar    | 60        | 2        | 208          |
| ORL      | 1024      | 40       | 400          |
| Colon    | 2000      | 2        | 62           |
| SRBCT    | 2308      | 4        | 83           |
| ARP      | 2400      | 10       | 130          |
| PIE      | 2420      | 10       | 210          |
| Leukemia | 7129      | 2        | 72           |
| CNS      | 7129      | 5        | 42           |

In the numerical experiments, the datasets are partitioned into the training and test sets. To minimize the effect of initial setting, ten-fold cross-validation mechanism [6], [7], [21], [31] is used for the simulations. The instances (along with their labels) of datasets are randomly divided into ten equal subsets to the extent possible, then one subset is selected as the test set and the union of the remaining nine subsets is used as the training set. The process rotates using each subset as the test set. The cross-validation experiment is repeated five times and the average accuracies as well as standard deviations are reported.

In order to verify the novelties of the proposed approach, we conduct the following three-part experiments. In Section IV-A, we first demonstrate the efficiency of the proposed gate function, (19), in learning importance of features. For this, we use the Wine dataset, as an example. In Section IV-B, we show the efficiency of the proposed Ada-softmin especially on high-dimensional classification problems. The robustness of the AdaTSK is investigated in supplementary materials. For this, we do not select features or reduce the size of the rule base. Then, in Section IV-C, we demonstrate how effective the proposed approach, FSRE-AdaTSK, is for feature selection as well as rule extraction for classification of the 19 datasets. We also compare the performance of FSRE-AdaTSK with a state-of-the-art method on 12 datasets. Finally, the computational complexity is discussed in Section IV-D.

A. Comparisons of Gate Functions

As mentioned in Section III-A, the proposed gate function, (19), can learn to distinguish between useful and poor features faster than the existing ones. To demonstrate this advantage, based on the feature selection method mentioned in Section III-C and AdaTSK using CoCo-FRB, the learning abilities of (19) and $M(\lambda) = e^{-x^2}$ are compared on the Wine dataset.

As explained in the computational protocol, we use three linguistic values, each modeled by a Gaussian membership function, for each feature. The system parameters are also initialized as we explained earlier. The $\lambda$s are initialized in such a manner that the gate values, $M(\lambda)$‘s, are initialized to values near 0.05 ($\lambda$s are around zero for the new gate function) to ensure that at the beginning all features are regarded as unimportant features. In the training procedure, the gate parameters are updated along with system parameters using full batch GD algorithm. The gate values over 1000 iterations are shown in Fig. 5. For Fig. 5(a) and (b), the learning coefficients are set to 0.01. As expected from Fig. 2(b), the gates modeled by (19) have opened faster and are scattered over a wide range [Fig. 5(a)] compared to the other gate function [Fig. 5(b)]. Consequently, the proposed gate function can distinguish between the useful and poor features faster than the other gate function. However, Fig. 5(b) may give a false impression that $M(\lambda) = e^{-x^2}$ cannot do its intended job. This is not true because either with a higher learning coefficient or with more iterations, $M(\lambda) = e^{-x^2}$ can also do its job. To demonstrate this, in Fig. 5(c) we depict the gate openings with the learning coefficient 0.05. However, since (19) has a significantly higher magnitude of its derivative with respect to $\lambda$ near the origin, for a given learning coefficient, it will exhibit a faster gate opening, which is an advantage.
TABLE III

| Dataset       | Product T-norm | Softmin \((q = -12)\) [5] | Ada-softmin (ours) |
|---------------|----------------|---------------------|-------------------|
| Iris          | 96.9           | 95.3                | 95.5              |
| Appendicitis  | 87.9           | 86.3                | 87.1              |
| Pima          | 76.4           | 75.4                | 76.1              |
| Yeast         | 54.0           | 48.9                | 55.8              |
| Glass         | 54.1           | 52.7                | 54.0              |
| Page-blocks   | 91.1           | 89.8                | 89.8              |
| Wine          | 98.8           | 98.6                | 98.7              |
| Heart         | 83.6           | 84.4                | 84.6              |
| Wdbc          | 96.9           | 94.2                | 94.3              |
| Texture       | 97.0           | 95.3                | 96.0              |
| Spect heartbeat| 80.4           | 78.9                | 79.8              |
| Sonar         | 75.3           | --                  | 75.1              |

Since the parameter of Ada-softmin, \(q\), can be adaptively acquired by (16) and the lower bound of \(q\) is set to 1000, the “numeric underflow” and “fake minimum” mentioned in Section II-B are avoided. As expected, the proposed AdaTSK can successfully deal with all of these datasets including seven high-dimensional datasets. In supplementary materials, we provide four figures to show the adaptation of the Ada-softmin.

C. Classification Performance of FSRE-AdaTSK

To demonstrate the efficient performance of FSRE-AdaTSK, we test it on the same 19 datasets with feature dimensionality varying from 4 to 7129. As explained earlier, there are three phases of the entire learning process. In the first phase, the feature selection phase, we use 10 fuzzy sets defined on every feature. The use of a large number of fuzzy sets is motivated by the fact that a high resolution fuzzy partition may help the feature selection process. We admit that for datasets like Iris, we do not need 10 fuzzy sets, but just to have a uniform policy for all datasets, we follow this. Moreover, in this phase we use the CoCo-FRB, i.e., use only \(R = S = 10\) rules. Thus, use of 10 rules will help model classes with complex structure. This will also indirectly reveal the robustness of the feature selection process. For the rule extraction phase, we define 5 linguistic values or fuzzy sets on each feature with a hope that since we already have identified useful features we may not need high resolution fuzzy partition. On the other hand, since in this phase we want to extract the final rules, we begin the process with more rules. In particular, in this phase, we use \(En-FRB\), i.e., we begin with \(R = (2D' + 1)S = (2D' + 1) \times 5\) rules, where \(D'\) is the number of selected features in the previous phase. Actually, these two values, 10 and 5, are some choices, not the only choices, that work and are by no means optimal. To choose the most desirable values of \(S\), one can use the cross-validation mechanism. To demonstrate that the choice of these parameters is not very critical, some experiments with different number of linguistic values on each feature are presented in supplementary materials.

For the feature selection phase, to make all features unimportant at the beginning of the training, the gate parameter, \(\lambda_{d_0}\), of every feature is initialized to 0.01, i.e., every gate value [19] is initialized to 0.0165. Similarly, at the beginning of the rule extraction phase, every rule gate parameter, \(\theta_{r_i}\), is set to 0.01. The initialization method of system parameters is identical to that described earlier in the beginning of Section IV. Since the feature selection and rule extraction are independent of each other, the system parameters and gate parameters for...
rule extraction must be reinitialized after the feature selection procedure. While in the fine tuning phase, all of the parameters do not need to be reinitialized. In our subsequent discussion we use $R'$ to denote the number of rules finally extracted from the second phase, which will be used in the final refinement phase.

As explained earlier, the antecedent parameters are not updated in the first two phases when handling with high-dimensional datasets. While the fine-tuning phase considers a simplified fuzzy rule base involving a reduced set of features and updates all of the system parameters including centers and consequent parameters. Ten-fold cross-validation mechanism is also used here and it is repeated five times as well. The average classification accuracy (Acc), average number of selected features ($#F$) and average number of extracted rules ($#R$) are reported in Tables IV and V for the low and high-dimensional datasets, respectively. Moreover, the standard deviations of the accuracies are also included. These two tables present only the test results. The training results (Tables S-II and S-III) and convergence plots (Fig. S-2) for some simulations along with discussions are provided in supplementary materials.

For low-dimensional datasets, in Table IV, we compare the results of FSRE-AdaTSK with those of a previous work [7], which is named FSCR-FRBS here. In [7], the authors used the gate function $M(\lambda) = e^{-\lambda^2}$ in a fuzzy rule-based framework, to select features with controlled redundancy. It is worth noting that the proposed FSRE-AdaTSK does not consider the problem controlling redundancy in the set of selected features. As seen from Table IV, the proposed scheme, FSRE-AdaTSK, uses fewer rules for most of the datasets and gets comparable results. Especially on some datasets, such as Appendicitis and Glass, FSRE-AdaTSK obtains higher classification accuracies using fewer features and rules.

For an easier comparison of performance measures in Fig. 6(a), corresponding to Table IV, we depict the number of features selected by FSCR-FRBS (blue bars) and FSRE-AdaTSK (orange bars) for the 12 datasets. Fig. 6(b), on the other hand, depicts the number of rules extracted by FSCR-FRBS (blue bars) and FSRE-AdaTSK (orange bars) for the same 12 datasets. Fig. 6(b) immediately reveals that for 10 of the 12 datasets FSRE-AdaTSK extracts fewer rules yielding better or comparable accuracies, which are depicted on the upper part of both Fig. 6(a) and (b). The blue curve corresponds to FSCR-FRBS and the orange curve corresponds to FSRE-AdaTSK. We note that the axis for the accuracies is shown on the right side of Fig. 6(a) and (b).

When we worked on this problem, we could not find any fuzzy rule-based systems to deal with feature selection and classification in an integrated manner that can deal with high-dimensional datasets, so we cannot compare the performance of FSRE-AdaTSK with any other fuzzy methods. However, in Table V we present results using two kinds of parameter estimation methods for FSRE-AdaTSK scheme. Different from aforementioned GD-based method, in the fine tuning phase, the consequent parameters are obtained using the least square error (LSE) estimation method while the antecedent parameters are kept fixed. The equation for the LSE estimate is not included here as this is not the primary research focus of this work. In [44], one can find the elaborate procedure for the LSE estimation of the consequent parameters. We provide two groups of results in Table V to demonstrate the effectiveness of our proposed approach on high-dimensional datasets. From Table V, it can be
concluded that FSRE-AdaTSK is capable of selecting features and extracting rules for high-dimensional problems and acquiring very satisfactory accuracies regardless of whether we use the LSE method or full batch GD algorithm in the fine tuning phase. From Tables IV and V we find that the standard deviation of accuracies over different runs for every dataset is very low even though there are three distinct phases of the algorithm. This clearly indicates the robustness of the system. Moreover, in order to show the robustness of the system with respect to the choice of number of linguistic values in the feature selection and rule extraction phases we have conducted experiments with different choices of the number of linguistic values and the performance of the algorithm is found to be very stable. These results are presented in Table S-I and discussed in supplementary materials.

Comparing the results of FSRE-AdaTSK reported in Tables IV and V with the results of normal TSK models reported in Table III, we observe that for 10 datasets, the FSRE-AdaTSK performs better than the conventional TSK model (with Ada-softmin). In fact, for some datasets, such as Leukemia, the improvement is even more than 13%. We note here that FSRE-AdaTSK uses a much smaller number of features than the standard neuro-fuzzy systems. Comparing Table III with Tables IV and V, we also observe that for several datasets, the performance of FSRE-AdaTSK is marginally lower than that of the standard TSK model. However, for two datasets, ORL and ARP, the performance of the standard neuro-fuzzy system is noticeably better than that of the FSRE-AdaTSK. A very plausible reason for this may be that the standard TSK model uses all features, while the FSRE-AdaTSK uses a small subset of features. And when the training dataset is too small to adequately represent the statistical characteristics of different classes, feature selection using a part of such data may not generalize well. This is possibly the main reason for such a drop in the performance of Ada-TSK for ORL and ARP.

Ada-softmin can adaptively acquire \( q \) according to the current membership values, which means the \( q_s \) are different in the different iterations. This adaptation ability of Ada-softmin is illustrated using the SRBCT dataset in Fig. S-3 and discussed in Section D of supplementary materials.

D. Computational Complexity

Here, we analyze the computational complexity of the proposed three-phase model for each instance. As \( R = S \) in the feature selection phase, the computational complexity of this step is \( O(DCS) \). For the rule extraction phase, \( R = (2D' + 1)S \), and its computational overhead is \( O(2DS + (D' + 2)R + 4R + (2D' + 1)RC + R + RC) \), i.e., \( O(D'^2CS) \). For the fine tuning phase, as the number of fuzzy rules is reduced from \( R \) to \( R' \), the computational cost is \( O(2DS + (D' + 2)R' + AR' + 2DR'RC + R' + RC) \), i.e., \( O(D'^2CR') \). As a consequence, the computational complexity for the forward propagation of FSRE-AdaTSK is \( O(DCS + D'^2CS + D'^2CR') \). Similarly, we can get the computational complexity for the back-propagation, which is also \( O(DCS + D'^2CS + D'^2CR') \) for each instance. Thus, taking into account the number of the instances, the overall complexity of the proposed three-phase algorithm is \( O(DCSN + D'^2CSN + D'^2CR'N) \).

As for the FSCR-FRBS method, the computational complexity of it is \( O(D^2 + D'^2 + R(D + D')) \), where \( R \) is the total number of rules. In [7], the training data from the \( c \)th class is clustered into \( n_c \) clusters and each cluster is converted into a rule. Thus, \( R = \sum_{c=1}^{C} n_c \), where \( n_c \) is the number of rules for the \( c \)th class. The authors in [7] used a fixed number of rules for each class, thus \( R = Cn_c \), where \( n_c = \hat{n}, \forall c = 1, 2, \ldots, C \). Since \( D > D' \), \( O(D^2 + D'^2 + R(D + D')) \) can be written as \( O(D^2 + DCn) \). Finally, the computational complexity of FSCR-FRBS with \( N \) input instances is \( O(D^2N + DCnN) \). We note here that, in reality, the training procedure of our proposed method is time consuming, especially for high-dimensional datasets, as there are three training phases. We next compare the complexity of a standard neuro-fuzzy system.

For the standard neuro-fuzzy system, \( R \times D \) membership values are calculated using (3), which requires a computational overhead of \( O(DRN) \). The cost of computing \( R \) firing strengths and their normalized values are \( O(DRN) \) and \( O(RN) \), respectively. For the computation of the consequents, the required computational complexity is \( O(DCRN) \), while that for the defuzzification process is \( O(CRN) \). Hence, the computational complexity for the back-propagation is also \( O(DCRN) \). Thus, the overall computational complexity of a standard neuro-fuzzy system is \( O(DCRN) \), which is lower than that of the proposed system. We note here that the complexity discussed above is for designing (training) the FSRE-AdaTSK and conventional neuro-fuzzy system. However, for any application of FSRE-AdaTSK, real-time or offline, the decision-making time is \( O(D'CR') \), while that for a standard neuro-fuzzy system is \( O(PCR) \), where \( D' < D \) and \( R' < R \). Thus in practice, for FSRE-AdaTSK, the actual decision-making time is much lower than that for the conventional neuro-fuzzy systems (without feature selection) because FSRE-AdaTSK uses significantly smaller number of features as well as smaller number of rules compared to its standard counterpart.

In addition, we also compare the actual computation time required by the proposed system with that of a standard neuro-fuzzy system empirically. These results are included in Table S-V and discussed in supplementary materials.

V. CONCLUSION

In this article, we propose a comprehensive fuzzy rule-based scheme called as FSRE-AdaTSK to solve high-dimensional classification problems. In order to address the problems associated with the computation of firing strength for high-dimensional data, we propose an adaptive softmin, called, Ada-softmin. The TSK model using Ada-Softmin is named here AdaTSK. The AdaTSK fuzzy system is capable of dealing with high-dimensional datasets. In addition, to facilitate feature selection and rule extraction using the AdaTSK framework, we propose a novel gate function, which is embedded into the system. This new gate function eliminates a limitation of the commonly used gate functions for feature selection. The feature selection and rule extraction are done in two different phases and for each phase we use a separate set of gate functions. The existing embedded methods for fuzzy rule-based feature selection use the gate function in the rule antecedents for the MA model and both in the antecedents and consequents for the TSK model. But the proposed system uses the gate function only with consequents both for feature selection and rule extraction. In the rule extraction/selection phase, we design a new type of fuzzy rule base, En-FRB, which begins with more rules than CoCo-FRB.
and avoids the exponential growth of the rules with increase in the number of input features.

We demonstrate the effectiveness of the proposed system on 19 datasets of which seven datasets have features from 1024 to 7129. To the best of our knowledge, this is the first time fuzzy rule-based systems have been designed involving data with more than 7000 features, where the feature selection is also done using a fuzzy rule-based framework. We note that both of the feature selection and rule extraction processes are based on the first-order TSK fuzzy system, so the number of system parameters that need to be optimized is quite large. In order to reduce the training burden, the zero-order TSK model will be considered in our future work. Use of the proposed AdaTSK framework for prediction/regression problems is straightforward. We plan to check the effectiveness of AdaTSK for function approximation/regression type problems. Besides, designing more efficient strategies for identifying fuzzy systems involving high-dimensional problems deserves further investigation.

REFERENCES

[1] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” Int. J. Man-Mach. Stud., vol. 7, no. 1, pp. 1–13, 1975.

[2] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control,” IEEE Trans. Syst., Man, Cybern., vol. SMC-15, no. 1, pp. 116–132, Jan./Feb. 1985.

[3] M. Sugeno and G. T. Kang, “Fuzzy modelling and control of multilayer incriminator,” Fuzzy Sets Syst., vol. 18, no. 3, pp. 329–345, 1986.

[4] D. Chakraborty 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, no. 3, pp. 391–400, Jun. 2001.

[5] 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, no. 1, pp. 110–123, Jan. 2004.

[6] 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.

[7] I.-F. Chung, Y.-C. Chen, and N. R. Pal, “Feature selection with controlled redundancy in a fuzzy rule based framework,” IEEE Trans. Fuzzy Syst., vol. 26, no. 2, pp. 734–748, Apr. 2018.

[8] T. Gao, Z. Zhang, Q. Chang, X. Xie, and J. Wang, “Conjugate gradient-based Takagi-Sugeno fuzzy neural network parameter identification and its convergence analysis,” Neurocomputing, vol. 364, pp. 168–181, 2019.

[9] D. Wu, Y. Yuan, J. Huang, and Y. Tan, “Optimize TSK fuzzy systems for regression problems: Minibatch gradient descent with regularization, DropRule, and AdaBound (MBGD-RDA),” IEEE Trans. Fuzzy Syst., vol. 28, no. 5, pp. 1003–1015, May 2020.

[10] Y. Cui, D. Wu, and Y. Xu, “Curse of dimensionality for TSK fuzzy neural networks: Explanation and solutions,” in Proc. Int. Joint Conf. Neural Netw., pp. 1–6, 2021.

[11] J. Wang, H. Zhang, J. Wang, Y. Pu, and N. R. Pal, “Feature selection using a neural network with group lasso regularization and controlled redundancy,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 3, pp. 1110–1123, Mar. 2021.

[12] S. Solorio-Fernández, J. F. Martínez-Trinidad, and J. A. Carrasco-Ochoa, “A new unsupervised spectral feature selection method for mixed data: A filter approach,” Pattern Recognit., vol. 72, pp. 314–326, 2017.

[13] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” Artif. Intell., vol. 97, no. 1, pp. 273–324, 1997.

[14] S. Maldonado and R. Weber, “A wrapper method for feature selection using support vector machines,” Int. J. Mach. Learn. Cybern., vol. 17, no. 1, pp. 2286–22874, 2018.

[15] M. M. Kabir, M. M. Islam, and K. Murase, “A new wrapper feature selection approach using neural network,” Neurocomputing, vol. 73, no. 16, pp. 3273–3283, 2010.

[16] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, “Hybrid feature selection by combining filters and wrappers,” Expert Syst. Appl., vol. 38, no. 7, pp. 8144–8150, 2011.

[17] J. Chen, T. Li, Y. Zou, G. Wang, H. Ye, and F. Lv, “An ensemble feature selection method for short-term electrical load forecasting,” in Proc. IEEE 3rd Conf. Energy Internet Energy Syst. Integration, 2019, pp. 1429–1432.

[18] Z. Hu, Y. Bao, T. Xiong, and R. Chiong, “Fuzzy hybrid wrapper feature selection for short-term load forecasting,” Eng. Appl. Artif. Intell., vol. 40, pp. 17–27, 2015.

[19] X.-Y. Liu, Y. Liang, S. Wang, Z.-Y. Yang, and H.-S. Ye, “A hybrid genetic algorithm with wrapper-embedded approaches for feature selection,” IEEE Access, vol. 6, pp. 22863–22874, 2018.

[20] D. Chakraborty and N. R. Pal, “Selecting useful groups of features in a connectionist framework,” IEEE Trans. Neural Netw., vol. 19, no. 3, pp. 381–396, Mar. 2008.

[21] R. Chakraborty and N. R. Pal, “Feature selection using a neural network model with controlled redundancy,” IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 1, pp. 35–50, Jan. 2015.

[22] H. Zhang, J. Wang, Z. Sun, J. M. Zarada, and N. R. Pal, “Feature selection for neural networks using group lasso regularization,” IEEE Trans. Knowl. Data Eng., vol. 32, no. 4, pp. 659–673, Apr. 2020.

[23] S. Abe and M.-S. Lan, “A method for fuzzy rules extraction directly from numerical data and its application to pattern classification,” IEEE Trans. Fuzzy Syst., vol. 3, no. 1, pp. 18–28, Feb. 1995.

[24] H. Ishibuchi, K. Nozaki, and H. Tanaka, “Distributed representation of fuzzy rules and its application to pattern classification,” Fuzzy Sets Syst., vol. 52, no. 1, pp. 21–32, 1992.

[25] 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, no. 3, pp. 260–270, Aug. 1995.

[26] H. Ishibuchi and T. Yamamoto, “Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining,” Fuzzy Sets Syst., vol. 141, no. 1, pp. 59–88, 2004.

[27] H. Ishibuchi, Y. Nojima, and I. Kuzawa, “Fuzzy data mining by heuristic rule extraction and multiobjective genetic rule selection,” in Proc. IEEE Int. Conf. Fuzzy Syst., 2006, pp. 1633–1640.

[28] N. Srinivasan, G. Hiron, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[29] L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, “Regularization of neural networks using DropConnect,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1058–1066.

[30] S. Feng and C. L. P. Chen, “Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification,” IEEE Trans. Cybern., vol. 50, no. 2, pp. 414–424, Feb. 2020.
[41] S. Gu, F.-L. Chung, and S. Wang, “A novel deep fuzzy classifier by stacking adversarial interpretable TSK fuzzy sub-classifiers with smooth gradient information,” IEEE Trans. Fuzzy Syst., vol. 28, no. 7, pp. 1369–1382, Jul. 2020.

[42] L.-X. Wang and J. Mendel, “Back-propagation fuzzy system as nonlinear dynamic system identifiers,” in Proc. IEEE Int. Conf. Fuzzy Syst., 1992, pp. 1409–1418.

[43] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” 2012, arXiv:1206.5533.

[44] N. R. Pal, V. K. Eluri, and G. K. Mandal, “Fuzzy logic approaches to structure preserving dimensionality reduction,” IEEE Trans. Fuzzy Syst., vol. 10, no. 3, pp. 277–286, Jun. 2002.

Guangdong Xue received the B.M. degree in logistics management from Hainan University, Haikou, China, in 2018. He is currently working toward the Ph.D. degree in control science and engineering with the China University of Petroleum (East China), Qingdao, China.

His current research interests include fuzzy systems, pattern recognition, and neural networks.

Qin Chang received the B.S. degree in applied mathematics from the China University of Petroleum (East China), Qingdao, China, in 2006, and the Ph.D. degree in probability and statistics from Shandong University, Jinan, China, in 2012.

In 2013, she joined the China University of Petroleum (East China), where she is currently a Lecturer with the Department of Data Science and Statistics, College of Science. Her current research interests include pattern recognition and classification, artificial neural networks, and bioinformatics.

Jian Wang (Senior Member, IEEE) received the B.S. degree in computational mathematics from the China University of Petroleum (East China), Qingdao, China, in 2002, and the M.S. and Ph.D. degrees in computational mathematics from the Dalian University of Technology, Dalian, China, in 2008 and 2012, respectively.

He is currently a Professor and serves as the Head of the Laboratory for Intelligent Information Processing with the College of Science, China University of Petroleum (East China). His research interests include computational intelligence, machine learning, pattern recognition, deep learning, differential programming, clustering, fuzzy systems, and evolutionary computation.

Prof. Wang was awarded several grants from the National Science Foundation of China, National Key Research and Development Program of China, Natural Science Foundation of Shandong Province, and Fundamental Research Funds for the Central Universities. He serves as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, International Journal of Machine Learning and Cybernetics, and Journal of Applied Computer Science Methods. He also serves on the Editorial Board of Neural Computing & Applications and Complex & Intelligent Systems. He served as a Guest Editor for Neural Computing & Applications and Computational Intelligence. In addition, he has served as the General Chair, the Program Chair, and the Co-Program Chair of several conferences such as the International Symposium on New Trends in Computational Intelligence, IEEE Symposium Series on Computational Intelligence and International Symposium on Neural Networks.

Kai Zhang received the Ph.D. degree in oil-gas field development engineering from the China University of Petroleum (East China), Qingdao, China, in 2008.

From June 2007 to May 2008, he studied in the University of Tulsa, Tulsa, OK, USA. In 2008, he joined as a Professor with the China University of Petroleum (East China), where he is also the Director of Talent Office. He teaches courses including fluid flow in porous media, reservoir engineering, etc. As a project leader, he has been in charge of three projects supported by the Natural Science Foundation of China (NSFC), one project supported by the National Natural Science Foundation of Shandong Province, one project supported by the National Ministry of Education, 20 projects supported by SINOPEC, CNOOC, and CNPC. He has published more than 60 papers. His research interests include reservoir simulation, production optimization, history matching, and development of unconventional reservoir.

Nikhil Ranjan Pal (Fellow, IEEE) is a Professor with the Electronics and Communication Sciences Unit of the Indian Statistical Institute (ISI), Kolkata, India. His current research interest includes brain science, computational intelligence, machine learning, and data mining.

Mr. Pal was the Editor-in-Chief of IEEE TRANSACTIONS ON FUZZY SYSTEMS for the period January 2005–December 2010. He has served/been serving on the editorial advisory board/steering committee of several journals including the International Journal of Approximate Reasoning, Applied Soft Computing, International Journal of Neural Systems, Fuzzy Sets and Systems, IEEE TRANSACTIONS ON FUZZY SYSTEMS, and IEEE TRANSACTIONS ON CYBERNETICS. He is a recipient of the 2015 IEEE Computational Intelligence Society (CIS) Fuzzy Systems Pioneer Award. He was a Distinguished Lecturer of the IEEE CIS (2010–2012, 2016–2018, 2022–2024) and was a Member of the Administrative Committee of the IEEE CIS (2010–2012). He has served as the Vice-President for Publications of the IEEE CIS (2013–2016) as well as the President of the IEEE CIS (2018–2019). He is a Fellow of the National Academy of Sciences, India, Indian National Academy of Engineering, Indian National Science Academy, International Fuzzy Systems Association (IFSA), and The World Academy of Sciences.