On the Functional Equivalence of TSK Fuzzy Systems to Neural Networks, Mixture of Experts, CART, and Stacking Ensemble Regression

Dongrui Wu, *Senior Member, IEEE*, Chin-Teng Lin, *Fellow, IEEE*, Jian Huang, *Senior Member, IEEE*, and Zhigang Zeng, *Senior Member, IEEE*

Abstract—Fuzzy systems have achieved great success in numerous applications. However, there are still many challenges in designing an optimal fuzzy system, e.g., how to efficiently optimize its parameters, how to balance the trade-off between cooperations and competitions among the rules, how to overcome the curse of dimensionality, how to increase its generalization ability, etc. Literature has shown that by making appropriate connections between fuzzy systems and other machine learning approaches, good practices from other domains may be used to improve the fuzzy systems, and vice versa. This article gives an overview on the functional equivalence between Takagi–Sugeno–Kang fuzzy systems and four classic machine learning approaches—neural networks, mixture of experts, classification and regression trees, and stacking ensemble regression—for regression problems. We also point out some promising new research directions, inspired by the functional equivalence, that could lead to solutions to the aforementioned problems. To our knowledge, this is so far the most comprehensive overview on the connections between fuzzy systems and other popular machine learning approaches, and hopefully will stimulate more hybridization between different machine learning algorithms.

Index Terms—CART, ensemble regression, fuzzy systems, mixture of experts, neural networks, stacking.

I. Introduction

RULE-BASED fuzzy systems have achieved great success in numerous applications [1], [2]. There are two kinds of rules for a fuzzy system: *Zadeh* [3], where the rule consequents are fuzzy sets, and *Takagi–Sugeno–Kang (TSK)* [4], where the rule consequents are functions of the inputs. Both types of fuzzy systems are universal approximators [5], [6].
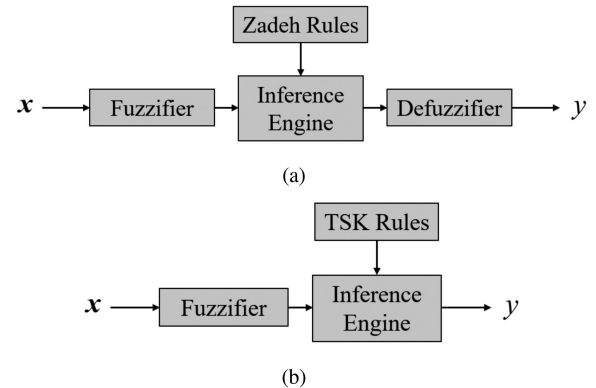
Fig. 1. (a) Flowchart of a Zadeh (Mamdani) fuzzy system. (b) Flowchart of a TSK fuzzy system.

As shown in Fig. 1(a), a Zadeh (Mamdani) fuzzy system consists of four components: *fuzzifier*, *rulebase*, *inference engine*, and *defuzzifier*. The fuzzifier maps each crisp input into a fuzzy set, the inference engine performs inferences on these fuzzy sets to obtain another fuzzy set, utilizing the Zadeh rules, and the defuzzifier converts the inferred fuzzy set into a crisp output. A TSK fuzzy system, shown in Fig. 1(b), does not need the defuzzifier, because the output of the inference engine is already a crisp number. Because of their simplicity and flexibility, TSK fuzzy systems are much more popular in practice. This article considers mainly TSK fuzzy systems for regression.

A TSK rule, as first proposed in [4], assumes the following general form:

$$\text{If } f(x_1 \text{ is } X_1, \ldots, x_d \text{ is } X_d), \text{ then } y = g(x_1, \ldots, x_d) \quad (1)$$

where $x_1, \ldots, x_d$ are inputs, $X_1, \ldots, X_d$ are membership functions (MFs), y is the output, f is a logical function connecting the antecedent propositions, and g is a function of the inputs.

In practice, simple TSK rules are often preferred. As an example, a TSK fuzzy system with d inputs and one output may have K rules in the following form:

$$R_k : \text{If } x_1 \text{ is } X_{k,1} \text{ and } \cdots \text{ and } x_d \text{ is } X_{k,d},$$

$$\text{then } y_k(\mathbf{x}) = \sum_{i=1}^{d} a_{k,i} x_i + b_k, \quad k = 1, \ldots, K$$

where $X_{k,i}$ is the MF for x_i in the kth rule, and $a_{k,i}$ and b_k are adjustable regression coefficients.

For a particular input $\mathbf{x} = (x_1, \ldots, x_d)$, the membership grade on $X_{k,i}$ is $\mu_{X_{k,i}}(x_i)$, and the firing levels of the rule are

$$f_k(\mathbf{x}) = \mu_{X_{k,1}}(x_1) \times \cdots \times \mu_{X_{K,d}}(x_d).$$

The output of the TSK fuzzy system is

$$
\begin{aligned}
y_{\text{TSK}}(\mathbf{x}) &= \frac{\sum_{k=1}^{K} f_k(\mathbf{x}) \cdot y_k(\mathbf{x})}{\sum_{k=1}^{K} f_k(\mathbf{x})} \\
&= \frac{\sum_{k=1}^{K} \left[f_k(\mathbf{x}) \cdot \left(\sum_{i=1}^{d} a_{k,i} x_i + b_k \right) \right]}{\sum_{k=1}^{K} f_k(\mathbf{x})}.
\end{aligned}
\tag{2}
$$

Or, if we define the normalized firing levels as

$$\bar{f}_k(\mathbf{x}) = \frac{f_k(\mathbf{x})}{\sum_{k=1}^{K} f_k(\mathbf{x})}, \quad k = 1, \ldots, K \tag{3}$$

then, (2) can be rewritten as

$$
\begin{aligned}
y_{\text{TSK}}(\mathbf{x}) &= \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}) \cdot y_k(\mathbf{x}) \\
&= \sum_{k=1}^{K} \left[\bar{f}_k(\mathbf{x}) \cdot \left(\sum_{i=1}^{d} a_{k,i} x_i + b_k \right) \right].
\end{aligned}
\tag{4}
$$

As pointed out in [7], there are many questions to be answered in designing a fuzzy system: Should singleton or nonsingleton fuzzifier be used? How many MFs should be used for each input? Should Gaussian or piecewise linear MFs be used? Should Mamdani or TSK inference be used? Should minimum or product t-norm be used? How to optimize the fuzzy system? In this article, we use singleton fuzzification, Gaussian MFs, TSK rules, and product t-norm, and assume that the user can specify the number of MFs in each input domain. We focus more on the optimization.

The challenges in optimizing a TSK fuzzy system include:

1) *Optimization:* A fuzzy system can be optimized by evolutionary algorithms [8], gradient descent [9], and gradient descent plus least squares estimation [10], as in the popular adaptive-network-based fuzzy inference system (ANFIS). However, evolutionary algorithms may be impractically slow on big data, gradient descent is very sensitive to the learning rate, and ANFIS can easily result in overfitting. So, it is necessary to develop more efficient and effective fuzzy system optimization algorithms, especially for big data.

2) *Interpretability:* A well-known advantage of fuzzy systems over many other machine learning approaches is the interpretability, i.e., one can look at each rule and understand how the fuzzy system is working. However, the interpretability decreases when the number of rules increases, and when each input activates too many rules. How to increase the interpretability, without sacrificing the learning performance, is another challenge.

3) *Curse of Dimensionality:* Fuzzy systems are particularly suffering from the curse of dimensionality. Assume a fuzzy system has d inputs, each with p MFs in its domain. Then, the total number of rules is p^d, i.e., the number of rules increases exponentially with the number of inputs, and the fuzzy system quickly becomes unmanageable. Clustering could be used to reduce the number of rules (one rule is extracted for each cluster) [11]–[14]. However, the validity of the clusters also decreases with the increase of feature dimensionality, especially when different features have different importance in different clusters [15], [16]. Additionally, the high dimensionality of features also increases the number of antecedents in the rules, and hence make them difficult to interpret.

4) *Generalization:* A fuzzy system should have not only good training performance, but also good generalization ability, i.e., it needs to perform well on the unknown test data. It is well known in machine learning that regularization can improve the generalization performance; however, the concept of regularization has not been extensively explored in training fuzzy systems.

This article gives a comprehensive overview of the functional equivalence[1] of TSK fuzzy systems to four classical machine learning algorithms: neural networks [18], mixture of experts (MoE) [19], classification and regression trees (CART) [20], and stacking ensemble regression [21]. Although a few publications on the connections of TSK fuzzy systems to some of these approaches have scattered in the literature, to our knowledge, no one has put everything together in one place so that the reader can easily see the big picture and get inspired. Moreover, we also discuss some promising hybridizations between TSK fuzzy systems and each of the four algorithms, which could be interesting new research directions. For example

1) By making use of the functional equivalence between TSK fuzzy systems and some neural networks, we can design more efficient training algorithms for TSK fuzzy systems.

2) By making use of the functional equivalence between TSK fuzzy systems and MoE, we may be able to achieve a better trade-off between cooperations and competitions of the rules in a TSK fuzzy system.

3) By making use of the functional equivalence between TSK fuzzy systems and CART, we can better initialize a TSK fuzzy system for high-dimensional problems.

4) Inspired by the connections between TSK fuzzy systems and stacking ensemble regression, we may be able to design better stacking models, and increase the generalization ability of a TSK fuzzy model.

The remainder of this article is organized as follows. Sections II–V describe the functional equivalence of TSK fuzzy systems to neural networks, MoE, CART, and stacking ensemble regression, respectively. Section VI draws conclusion.

---

[1]It has been shown that many machine learning algorithms are universal approximators [6], [17]. However, both algorithms being universal approximators does not mean that they are functionally equivalent: universal approximation usually requires a very large number of nodes or parameters, so it is theoretically important, but may not be used in real-world algorithm design. By functional equivalence, we emphasize that two algorithms can implement *exactly* the same function with a relatively *small* number of parameters.
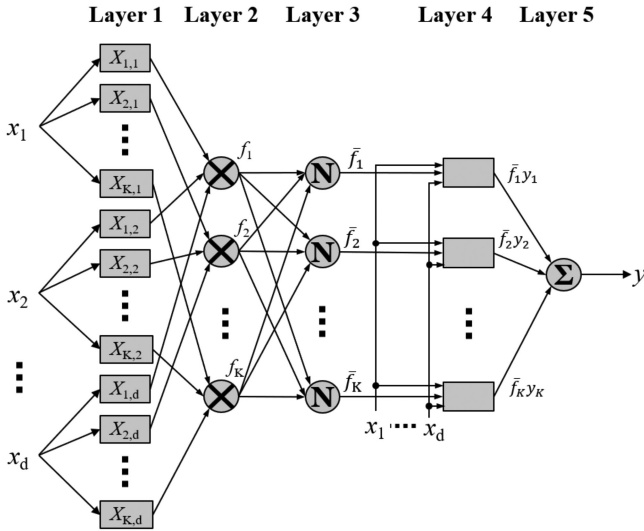
Fig. 2. TSK fuzzy system introduced in Section I, represented as a five-layer ANFIS.

II. TSK FUZZY SYSTEMS AND NEURAL NETWORKS

Neural networks have a longer history[2] than fuzzy systems, and are now at the center stage of machine learning, because of the booming of deep learning [22].

Researchers started to discover in the early 1990s that a TSK fuzzy system can be represented similarly to a neural network [9], [10], [23]–[25], so that a neural network learning algorithm, such as backpropagation [18], can be used to train it. These fuzzy systems are called *neuro-fuzzy systems* in the literature [2].

A. ANFIS

Among the many variants of neuro-fuzzy systems, the most popular one may be the ANFIS [10], which has been cited over 15,000 times on Google Scholar, and implemented in the MATLAB Fuzzy Logic Toolbox. The ANFIS structure of the d-input one-output TSK fuzzy system, introduced in Section I, is shown in Fig. 2. It has five layers.

1) *Layer 1:* The membership grade of x_i on $X_{k,i}$ ($k = 1, \ldots, K$; $i = 1, \ldots, d$) is computed.
2) *Layer 2:* The firing level of each rule R_k is computed, by multiplying the membership grades of the corresponding rule antecedents.
3) *Layer 3:* The normalized firing levels of the rules are computed, using (3).
4) *Layer 4:* Each normalized firing level is multiplied by its corresponding rule consequent.
5) *Layer 5:* The output is computed by (4).

All parameters of the ANFIS, i.e., the shapes of the MFs and the rule consequents, can be trained by a gradient descent algorithm [10]. Or, to speed up the training, a more efficient hybrid learning algorithm [10] can be used. In the forward
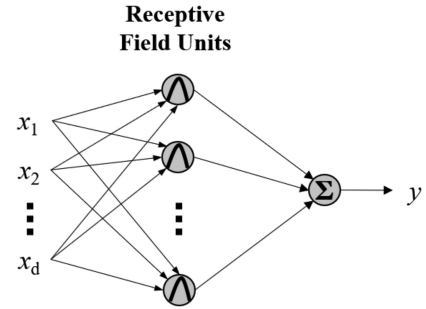
pass, the antecedent parameters are fixed, the functional signals go forward till Layer 4, and the consequent parameters are optimized by least squares estimation. In the backward pass, the consequent parameters are fixed, the errors propagate backward, and the antecedent parameters are updated by gradient descent.

B. Functional Equivalence Between TSK Fuzzy Systems and Radial Basis Functional Networks (RBFN)

A variant of neural networks, the radial basis function network (RBFN) [26], which is a universal approximator [27], is functionally equivalent to a TSK fuzzy system under certain constraints. A radial basis function is a real-valued function whose value depends only on the distance from a center $\mathbf{c}$, i.e., $f(\mathbf{x}, \mathbf{c}) = f(\|\mathbf{x} - \mathbf{c}\|)$.

An RBFN [26] uses local receptive fields, inspired by biological receptive fields, for function mapping. Its diagram is shown in Fig. 3. For an input $\mathbf{x} = (x_1, \ldots, x_d)$, the output of the kth ($k = 1, \ldots, K$) receptive field unit, using a Gaussian response function, is

$$f_k(\mathbf{x}) = \exp\left(-\frac{\sum_{i=1}^{d}(x_i - c_{k,i})^2}{\sigma_k^2} \right) \qquad (5)$$

where $c_{k,i}$ is the center of the Gaussian function for x_i, and σ_k is the common standard deviation of the Gaussian functions.

With the addition of lateral connections (not shown in Fig. 3) between the receptive field units, the output of the (normalized) RBFN is

$$y(\mathbf{x}) = \frac{\sum_{k=1}^{K} f_k(\mathbf{x}) \cdot y_k}{\sum_{k=1}^{K} f_k(\mathbf{x})} \qquad (6)$$

where y_k is a constant output associated with the kth receptive field unit.[3]

Jang and Sun [29] have shown that a TSK fuzzy system [see (4)] is functionally equivalent to an RBFN [see (6)], if the following constraints are satisfied.

1) The number of receptive field units equals the number of fuzzy rules.

Fig. 3. RBFN.

---

[2][Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

[3]There is a related machine learning approach called local model networks [28], which can be viewed as a decomposition of a complex nonlinear system into a set of locally accurate submodels smoothly integrated by their associated basis functions. It replaces the constant output of each receptive unit in an RBFN by a function of the inputs.

2) The output of each fuzzy rule is a constant, instead of a function of the inputs.
3) The antecedent MFs of each fuzzy rule are Gaussian functions with the same variance.
4) The product t-norm is used to compute the firing level of each rule.
5) The fuzzy system and the RBFN use the same method (i.e., either weighted average or weighted sum) to compute the final output.

Actually, the Gaussian function requirement in Constraint (3) may not be necessary. When the other four constraints are satisfied, as long as the MFs in the TSK fuzzy system are in the same form as the radial basis functions (not necessarily Gaussian) in the RBFN, the TSK fuzzy system and the RBFN are equivalent.

Hunt *et al.* [30] proposed a generalized RBFN, which has the following main features, compared with the above standard RBFN.
1) A receptive field unit may be connected with only a subset of the inputs, instead of all inputs in the standard RBFN.
2) The output associated with each receptive field unit can be a linear or nonlinear function of the inputs, instead of a constant in the standard RBFN.
3) The Gaussian response functions of the receptive field units can have different variances for different inputs, instead of identical variance in the standard RBFN.

Then, the generalized RBFN is functionally equivalent to a TSK fuzzy system, under the following constraints [30].
1) The number of receptive field units equals the number of fuzzy rules.
2) The antecedent MFs of each fuzzy rule are Gaussian.
3) The product t-norm is used to compute the firing level of each rule.
4) The fuzzy system and the RBFN use the same method (i.e., either weighted average or weighted sum) to compute the final output.

Again, the Gaussian constraint can be relaxed to other radial basis functions.

The training of an RBFN consists of two steps.
1) Determine the center vectors of the RBFs in the hidden layer. They can be initialized randomly, or through clustering, or by grid partition of the input space.
2) Fit a linear model to the hidden layer's outputs according to some loss function, e.g., the least squares loss in regression.

Because of the functional equivalence between an RBFN and a TSK fuzzy system, the techniques used in Step (1) above can also be used to initialize the MFs in a TSK fuzzy systems. For example, in evolutionary fuzzy systems design [8], the input MFs are randomly initialized, and a fitness function is used to select the configuration that achieves the best performance. In Yen *et al.*'s approach [31], to increase the interpretability of a TSK fuzzy system, the antecedent fuzzy partitions are determined by starting with an oversized number of partitions, and then removing redundant and less important ones using the SVD-QR algorithm [32]. There have also been many approaches for generating initial fuzzy rule partitions through clustering [11]–[14].

Different clustering algorithms, e.g., mountain clustering [11], fuzzy c-means clustering [12], aligned clustering [14], etc., have been used.

Moreover, some efficient and effective training approaches for RBFN have been proposed recently, which may also be extended to TSK fuzzy system. For example, multicolumn RBFN [33], which divides a large dataset into smaller subsets using the k-d tree algorithm and then trains an RBFN for each subset, has demonstrated faster speed and higher accuracy than the traditional RBFN. This approach could be extended to TSK fuzzy systems for big data problems.

C. Discussions and Future Research

As ANFIS is an efficient and popular training algorithm for type-1 TSK fuzzy systems, it is natural to consider whether it can also be used for interval and general type-2 fuzzy systems [1], which have demonstrated better performance than type-1 fuzzy systems in many applications. There have been limited research in this direction [34], [35]. Unfortunately, it was found that interval type-2 ANFIS may not outperform type-1 ANFIS. One possible reason is that when the Karnik–Mendel algorithms [1] are used in type-reduction of the interval type-2 fuzzy system, the least squares estimator in the interval type-2 ANFIS does not always give the optimal solution, due to the switch point mismatch [34]. A remedy may be to use an alternative type-reduction and defuzzification approach [36], which does not involve the switch points, e.g., the Wu–Tan method [37]. This is a direction that we are currently working on.

Many novel approaches have been proposed in the last few years to speed up the training and increase the generalization ability of deep neural networks, e.g., Dropout [38], dropConnect [39], and batch normalization [40]. Dropout randomly discards some neurons and their connections during the training. DropConnect randomly sets some connection weights to zero during the training. Batch normalization normalizes the activation of the hidden units, and hence reduces internal covariate shift.[4] Similar concepts may also be used to expedite the training and increase the generalization ability of TSK fuzzy systems. For example, inspired by Dropout, we recently developed a novel dropRule approach [41] for training TSK fuzzy systems for regression problems, which drops some rules randomly in each epoch of the training. A batch normalization approach has also been proposed for training TSK fuzzy systems for classification problems [42].

Although deep learning has achieved great success in numerous applications, its model is essentially a black-box because it is difficult to explain the acquired knowledge or decision rationale. This may hinder it from safety-critical applications such as medical diagnoses. Explainability of deep learning models has attracted a rapidly growing research interest in the past few years. According to Amarasinghe and Manic, there have been

---

[4]As explained in [40], internal covariate shift means "the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities."
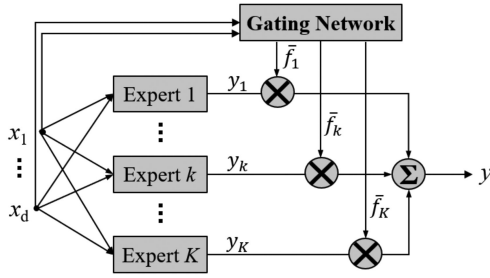
Fig. 4.   Mixture of experts (MoE).

two groups of research on this [43]: 1) altering the learning algorithms to learn explainable features; and, 2) using additional methods with the standard learning algorithm to explain existing deep learning algorithms. They [43] also presented an interesting methodology for linguistically explaining the knowledge a deep neural network classifier has acquired in training, using linguistic summarization [44], which generates Zadeh fuzzy rules. This work shows a novel and promising application of fuzzy rules in deep learning. Similarly, TSK fuzzy rules could also be used to linguistically explain a deep regression model.

Finally, fuzzy logic and deep learning could be hybridized to take the advantages of both sides [45], [46]. Deng *et al.* [45] proposed a hierarchical deep neural network that derives information from both fuzzy and neural representations, which are then fused to form the final features for classification. Zheng *et al.* [46] proposed a Pythagorean fuzzy deep Boltzmann machine, in which the deep Boltzmann machine parameters are expressed by Pythagorean fuzzy numbers so that each neuron can learn how a feature affects the output both positively and negatively. We expect that more hybridizations like these will emerge in the near future.

III. TSK FUZZY SYSTEMS AND MoE

MoE was first proposed by Jacobs *et al.* in 1991 [19]. It is established based on the divide-and-conquer principle, in which the problem space is divided among multiple local experts, supervised by a gating network, as shown in Fig. 4. MoE trains multiple local experts, each taking care of only a small local region of the problem space; for a new input, the gating network determines which experts should be used for it, and then aggregates the outputs of these experts by a weighted average. MoE models are universal approximators [47].

A. MoE

Assume there are N training examples $(\mathbf{x}_n, y_n)$, $n = 1, \ldots, N$. The K experts are trained from minimizing the following error:[5]

$$E = \sum_{n=1}^{N} \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}_n)(y_n - y_k(\mathbf{x}_n))^2 \tag{7}$$

---

[5]In practice, transforms of (7), e.g., $E = -\log \sum_{n=1}^{N} \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}_n) \exp\left[-\frac{1}{2}(y_n - y_k(\mathbf{x}_n))^2\right]$, may be used to speed up the optimization [19].

where $y_k(\mathbf{x}_n)$ is the output of the kth expert for input $\mathbf{x}_n$, and $\bar{f}_k(\mathbf{x}_n)$ is the corresponding normalized weight for the kth expert, assigned by the gating network

$$\bar{f}_k(\mathbf{x}_n) = \frac{\exp(f_k(\mathbf{x}_n))}{\sum_{i=1}^{K} \exp(f_i(\mathbf{x}_n))} \tag{8}$$

in which $f_k(\mathbf{x}_n)$ is a tunable function.

Once the training is done, the final output of the MoE is

$$y_{MoE}(\mathbf{x}) = \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}) y_k(\mathbf{x}). \tag{9}$$

B. Functional Equivalence Between TSK Fuzzy Systems and MoE

It is easy to see that the TSK fuzzy system in (4) and the MoE in (9) are conceptually equivalent. More specifically, when the following conditions are satisfied, they are functionally equivalent, i.e., (4) is identical to (9).

1) The TSK fuzzy system uses Gaussian MFs and the product t-norm. Assume the Gaussian MF $X_{k,i}$ of the TSK fuzzy system [see (1)] has center $c_{k,i}$ and standard deviation $\sigma_{k,i}$. Then, $f_k(\mathbf{x}_n)$ in (8) of the MoE should be

$$f_k(\mathbf{x}_n) = -\sum_{i=1}^{d} \frac{(x_{n,i} - c_{k,i})^2}{2\sigma_{k,i}^2}. \tag{10}$$

2) $y_k(\mathbf{x})$ in the TSK fuzzy system is identical to $y_k(\mathbf{x})$ in the MoE, for every k.

A few publications [48], [49] have made the connection between TSK fuzzy systems and MoE. The regression function in each rule consequent of the TSK fuzzy system can be viewed as an expert, and the rule antecedents work as the gating network: for each input, they determine how much weight should be assigned to each rule consequent (expert) in the final aggregation. Of course, the experts and gating network in MoE can be constructed by more complex models, such as neural networks and support vector machines [50], but the structure resemblance remains unchanged.

C. Discussions and Future Research

Lots of progresses on MoE have been made since it was first proposed in 1991 [50], [51], e.g., different training algorithms, different gating networks, and different expert models. Since MoE is essentially identical to a TSK fuzzy system, these ideas could also be applied to fuzzy systems, particularly the training algorithms and expert models (the gating network is a little more challenging because in a TSK fuzzy system, we always use MFs to perform gating; there is not too much freedom).

First, in training a TSK fuzzy system for regression, the error function is usually defined as

$$E = \sum_{n=1}^{N} (y_n - y_{\text{TSK}}(\mathbf{x}_n))^2$$

$$= \sum_{n=1}^{N} \left[y_n - \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}_n) y_k(\mathbf{x}_n) \right]^2. \tag{11}$$

However, as pointed out in [19], "this error measure compares the desired output with a blend of the outputs of the local experts, so, to minimize the error, each local expert must make its output cancel the residual error that is left by the combined effects of all the other experts. When the weight in one expert change, the residual error changes, and so the error derivatives for all other local experts change." This strong coupling between the experts facilitates their cooperation, but may lead to a solution in which many experts are used for each input. That is why in training the local experts, the error function is defined as (7) to facilitate the competition among them. Equation (7) requires each expert to approximate y_n, instead of a residual. Hence, each local expert is not directly affected by other experts (it is indirectly affected by other experts through the gating network, though).

It is thus interesting to study if changing the error function from (11) to (7) in training a TSK fuzzy system can improve its performance, in terms of speed and accuracy. Or, the error function could be a hybridization of (11) and (7), to facilitate both the cooperations and competitions among the local experts, i.e.,

$$E = \sum_{n=1}^{N} \left[y_n - \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}_n) y_k(\mathbf{x}_n) \right]^2 + \lambda \sum_{n=1}^{N} \sum_{k=1}^{K} \bar{f}_k(\mathbf{x}_n) \left[y_n - y_k(\mathbf{x}_n) \right]^2 \quad (12)$$

where λ is a hyperparameter defining the trade-off between cooperation and competition. This idea was first explored in [31], which proposed a TSK fuzzy system design strategy to increase its interpretability, by forcing each rule consequent to be a reasonable local model (the regression function in each rule consequent needs to fit the training data that are covered by the rule antecedent MFs well), and also the overall TSK fuzzy system to be a good global model. However, the algorithms in [31] are very memory-hungry,[6] and hence may not be applicable when N is large. A more efficient solution to this problem is needed.

Second, when the performance of an initially designed TSK fuzzy system is not satisfactory, there could be two strategies to improve it: 1) increase the number of rules, so that each rule covers a smaller region in the input domain, and hence may better approximate the training examples in that region; and, 2) increase the fitting power (nonlinearity) of the consequent function, so that it can better fit the training examples in its local region. The first strategy is frequently used in practice; however, it can increase the number of parameters of the TSK fuzzy system very rapidly. Juang and Lin [14] proposed an interesting approach to incrementally add linear terms to the rule consequent to increase its fitting power. However, they only considered linear terms. Inspired by MoE, whose expert models could use complex models like the neural networks and support vector machines [50], the TSK rule consequents (local experts) could also use more sophisticated models, particularly,
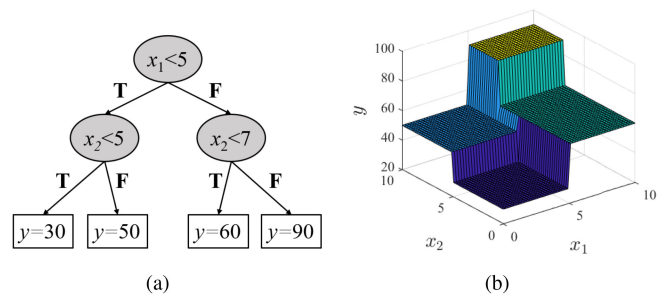

Fig. 5. (a) Example of CART for regression; and (b) its input–output mapping.

support vector regression [52], which outperforms simple linear regression in many applications. The feasibility of this idea has been verified in [53], [54].

Third, TSK fuzzy rules could also be used as experts in MoE. For example, Leski [55] proposed such an approach for classification: each expert model in the MoE was constructed as a TSK fuzzy rule (whose input region was determined by fuzzy c-means clustering), and then a gating network was used to aggregate them. This may increase the interpretability of MoE. This idea can also be extended to regression problems.

IV. TSK FUZZY SYSTEMS AND CART

CART [20] is a popular and powerful strategy for constructing classification and regression trees. It is a universal approximator [56], and has also been used in ensemble learning such as random forests [57] and gradient boosting machines [58]. This section focuses on regression only.

A. CART

Assume there are two numerical inputs, x_1 and x_2, and one output, y. An example of CART is shown in Fig. 5(a). It is constructed by a divide-and-conquer strategy, in which the input space is partitioned by a hierarchy of Boolean tests into multiple nonoverlapping partitions. Each Boolean test corresponds to an internal node of the decision tree. The leaf node (terminal node) is computed as the mean y of all training examples falling into the corresponding partition; thus, CART implements a piecewise constant regression function, as shown in Fig. 5(b). The route leading to each leaf node can be written as a crisp rule, e.g., if $x_1 < 5$ and $x_2 < 5$, then $y = 30$. Note that each leaf node can also be a function of the inputs [59]–[63], instead of a constant. In this way, the implemented regression function is smoother; however, the trees are more difficult to train.

B. Functional Equivalence Between TSK Fuzzy Systems and CART

Both CART and fuzzy systems use rules. The rules in CART are crisp: each input belongs to only one rule, and the output is the leaf node of that rule. On the contrary, the rules in a fuzzy system are fuzzy: each input may fire more than one rules, and the output is a weighted average of these rule consequents.

---

[6]Let N be the number of training examples, r the dimensionality of the input, and L the number of rules. The algorithms in [31] need to construct matrices in $\mathbb{R}^{NL \times L(r+1)}$ and $\mathbb{R}^{NL \times NL}$, which are hardly scalable.
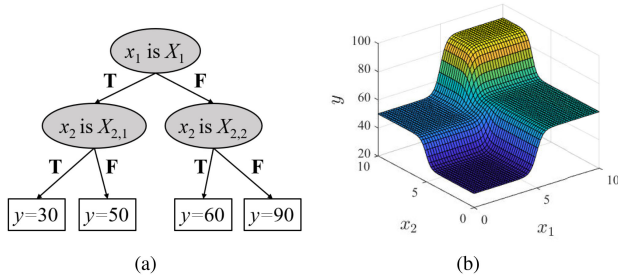
Fig. 6. (a) An example of fuzzy CART for regression; and (b) its input–output mapping.

The regression output of a traditional CART has discontinuities, which may be undesirable in practice. So, fuzzy CART, which allows an input to belong to different leaf nodes with different degrees, has been proposed to accommodate this [64]–[67]. As pointed out by Suarez and Lutsko [67], "in regression problems, it is seen that the continuity constraint imposed by the function representation of the fuzzy tree leads to substantial improvements in the quality of the regression and limits the tendency to overfitting." An example of fuzzy CART for regression is shown in Fig. 6(a), where X_1 is a fuzzy set for x_1, and $X_{2,1}$ and $X_{2,2}$ are fuzzy sets for x_2. Its input–output mapping is shown in Fig. 6(b), which is continuous.

Let the kth constant leaf node in a fuzzy CART be y_k. Then, given an input $\mathbf{x}$, the output of a fuzzy CART is a weighed average of the predictions at all leaves

$$y_{\text{fCART}}(\mathbf{x}) = \frac{\sum_{k=1}^{K} f_k(\mathbf{x})y_k}{\sum_{k=1}^{K} f_k(\mathbf{x})} \tag{13}$$

where $f_k(\mathbf{x})$ is the product of all membership grades in the path to y_k. In this way, $y_{fCART}(\mathbf{x})$ is a smooth function. Clearly, $y_{\text{fCART}}(\mathbf{x})$ is functionally equivalent to the output of a TSK fuzzy system in (2), when each rule consequent of the fuzzy system is a constant (instead of a function).

Chaudhuri *et al.* [61] proposed smoothed and unsmoothed piecewise-polynomial regression trees (SUPPORT), in which each leaf node is a polynomial function of the inputs. The SUPPORT tree is generally much shorter than a traditional CART tree, and hence enjoys better interpretability. The following three-step procedure is used to ensure that its output is smooth.

1) The input space is recursively partitioned until the data in each partition are adequately fitted by a fixed-order polynomial. Partitioning is guided by analyzing the distributions of the residuals and the cross-validation estimates of the mean squared prediction error.
2) The data within a neighborhood of the kth ($k = 1, \ldots, K$) partition are fitted by a polynomial $y_k(\mathbf{x})$.
3) The prediction for an input $\mathbf{x}$ is a weighted average of $y_k(\mathbf{x})$, where the weighting function $f_k(\mathbf{x})$ diminishes rapidly to zero outside the kth partition.

If the weighting functions are Gaussian-like, i.e.,

$$f_k(\mathbf{x}) = \exp\left(-\frac{(x_1 - m_{k,1})^2}{\sigma_{k,1}^2} - \frac{(x_2 - m_{k,2})^2}{\sigma_{k,2}^2}\right) \tag{14}$$

where $m_{k,i}$ is the mean of the Gaussian function for the ith input, and $\sigma_{k,i}$ is the standard deviation, then the output of

SUPPORT is

$$y_{\text{SUPPORT}}(\mathbf{x}) = \frac{\sum_{k=1}^{K} f_k(\mathbf{x})y_k(\mathbf{x})}{\sum_{k=1}^{K} f_k(\mathbf{x})}. \tag{15}$$

Clearly, $y_{\text{SUPPORT}}(\mathbf{x})$ is functionally equivalent to the output of the TSK fuzzy system in (2).

C. Discussions and Future Research

It is well known that fuzzy systems are particularly subject to the curse of dimensionality: as the number of features increases, the number of rules may increase exponentially, and the interpretability also decreases quickly as the number of antecedents increases.

CART may offer a solution to this problem. For example, Jang [65] first performed CART on a regression dataset to roughly estimate the structure of a TSK fuzzy system, i.e., number of MFs in each input domain, and the number of rules. Then, each crisp rule antecedent was converted into a fuzzy set, and consequent to a linear function of the inputs. For example, a crisp rule

$$R_k : \text{ If } x_1 > x_{k,1} \text{ and } x_2 < x_{k,2}, \text{ then } y_k = c_k$$

can be converted to a TSK fuzzy rule

$$R_k : \text{ If } x_1 \text{ is } X_{k,1} \text{ and } x_2 \text{ is } X_{k,2}$$
$$\text{then } y_k = a_k x_1 + b_k x_2 + c'_k$$

where a_k, b_k, and c'_k are regression coefficients, and $X_{k,1}$ and $X_{k,2}$ are fuzzy sets defined as

$$\mu_{X_{k,1}}(x_1) = \frac{1}{1 + \exp[-\alpha_{k,1}(x_1 - x_{k,1})]} \tag{16}$$

$$\mu_{X_{k,2}}(x_2) = \frac{1}{1 + \exp[\alpha_{k,2}(x_2 - x_{k,2})]} \tag{17}$$

in which $\alpha_{k,1}$ and $\alpha_{k,2}$ are tunable parameters. Once all crisp rules have been converted to TSK fuzzy rules, ANFIS [10] can be used to optimize the parameters of all rules together, e.g., a_k, b_k, c'_k, $\alpha_{k,1}$, and $\alpha_{k,2}$.

The above TSK fuzzy system design strategy offers at least three advantages.

1) *Simplicity:* We can prune the CART tree on a high-dimensional dataset to obtain a regression tree with a desired number of leaf nodes, and hence a TSK fuzzy system with a desired number of rules. So, we can directly control the simplicity of the resulted TSK fuzzy system.
2) *Interpretability:* Rules in a traditionally designed fuzzy system usually have the same number of antecedents (which equals the number of inputs), which are difficult to interpret when there are many antecedents. Rules initialized from CART may have different number of antecedents (which are usually smaller than the number of inputs), depending on the depths of the corresponding leaf nodes, i.e., we can extract shorter and more interpretable rules that may not be extractable using a traditional fuzzy system design approach.
3) *Performance:* In a traditional fuzzy system, each input (feature) is independently considered in rule antecedents.
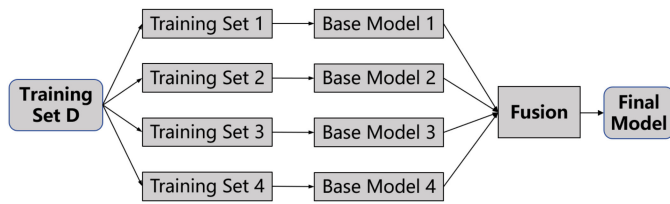
Fig. 7. Stacking ensemble regression.

However, some variants of CART [59] allow to split on the linear combinations of the inputs, which is equivalent to using new (usually more informative) features in splitting. These new features are also used by fuzzy rules when they are converted from CART leaf nodes, which may achieve better performance than traditional fuzzy systems.

In summary, initializing TSK fuzzy systems from CART regression trees is a promising solution to high-dimensional problems, and may achieve better interpretability-performance trade-off. Hence, it deserves further research.

V. FUZZY SYSTEM AND STACKING

Ensemble regression [21] improves the regression performance by integrating multiple base models. Stacking may be the simplest supervised ensemble regression approach. Its final regression model is a weighted average of the base models, where the weights are trained from the labeled training data. As long as the base models are universal approximators, the stacking model should also be a universal approximator.

A. Stacking

The base models in stacking may be trained from other related tasks or datasets [68]. However, when there are enough training examples for the problem under consideration, the base models may also be trained directly from them. Fig. 7 illustrates such a strategy. For a given training dataset, we can resample (e.g., using bootstrap [69]) it multiple times to obtain multiple new training datasets, each of which is slightly different from the original training dataset. Then, a base model can be trained using each resampled dataset. These base models could use the same regression algorithm, but different regression algorithms, e.g., LASSO [70], ridge regression [71], support vector regression [52], etc., can also be used to increase their diversity. Because the training datasets are different, the trained base models will be different even if the same regression algorithm is used.

Once the K base models are obtained, stacking trains another (linear or nonlinear) regression model to fuse them. Assume the outputs of the K base regression models are $\{y_k\}_{k=1}^K$. Stacking finds a regression model $y = f(y_1, \ldots, y_K)$ on the training dataset to aggregate them.

B. Connections Between TSK Fuzzy Systems and Stacking

A TSK fuzzy system for regression can be viewed as a stacking model. Each rule consequent is a base regression model, and the rule antecedent MFs determine the weights of the base models in stacking. Note that in stacking, usually the aggregated output y is a function of $\{y_k\}_{k=1}^K$ only, but in a TSK fuzzy system, the aggregation function also depends on the input $\mathbf{x}$, as the weights are computed from them, and change with them. So, a TSK fuzzy system is actually an adaptive stacking regression model.

A key issue in stacking is the partition of the original training dataset, after which a base regression model can be built from each partition. Two commonly used data partition approaches are: 1) bootstrap sampling, where each base model is trained from about 63.2% unique samples of the original training dataset; and 2) k-fold partition, where each of the k base models is trained from $\frac{100(k-1)}{k}\%$ samples of the original training dataset. Similar dataset partition concepts have also been used to construct or initialize TSK fuzzy systems; however, they may be better interpreted from a possibility point of view, where the possibility (weight) of a sample in constructing a TSK rule is usually a function of the corresponding firing level.

For example, Nozaki *et al.* [72] proposed a simple yet powerful heuristic approach for generating TSK fuzzy rules (whose rule consequents are constants, instead of functions of the inputs) from numerical data. Assume there are N training examples $(x_{n,1}, \ldots, x_{n,d}, y_n)$, $n = 1, \ldots, N$, i.e., the fuzzy system has d inputs and one output. Then, Nozaki *et al.*'s approach consists of the following steps [72].

1) Determine how many MFs should be used for each input, and define the shapes of the MFs. Once this is done, the input space is partitioned into several fuzzy regions.

2) Generate a fuzzy rule in the form of

$$R_k : \text{If } x_1 \text{ is } X_{k,1} \text{ and } \cdots \text{ and } x_d \text{ is } X_{k,d},$$
$$\text{then } y_k = c_k$$

in the kth fuzzy region, where the MFs $X_{k,i}$ $(i = 1, \ldots, d)$ have been determined in Step (1), and

$$c_k = \frac{\sum_{n=1}^N \left[\prod_{i=1}^d \mu_{X_{k,i}}(x_{n,i}) \right]^\alpha y_n}{\sum_{n=1}^N \left[\prod_{i=1}^d \mu_{X_{k,i}}(x_{n,i}) \right]^\alpha} \quad (18)$$

in which α is a positive constant. c_k could also be computed using a least squares approach [72].

Essentially, the above rule-construction approach reweights each training example in a fuzzy partition using an exponential function of the firing level of the corresponding rule, and then computes a simple base model $y_k = c_k$ from them. The final TSK fuzzy system is an aggregation of all such rules. This is exactly the idea of stacking in Fig. 7.

Another example is the local learning part in [31], where an approach for constructing a local TSK rule for each rule partition is proposed. Using again the d-input one-output example in Section 1, a local TSK rule is in the form of

$$R_k : \text{If } x_1 \text{ is } X_{k,1} \text{ and } \cdots \text{ and } x_d \text{ is } X_{k,d}$$
$$\text{then } y_k = \sum_{i=1}^d a_i x_i + b_k.$$

Given $X_{k,i}$ $(i = 1, \ldots, d)$, the weight for each training example $(x_{n,1}, \ldots, x_{n,d}, y_n)$ is $f_k(\mathbf{x}_n) = \prod_{i=1}^{d} \mu_{X_{k,i}}(x_{n,i})$, i.e., its firing level of the rule, and then the regression coefficients $a_{k,i}$ and b_k are found from minimizing the following weighted loss:

$$E = \sum_{n=1}^{N} f_k(\mathbf{x}_n) \left[y_n - \left(\sum_{i=1}^{d} a_{k,i} x_{n,i} + b_k \right) \right]^2. \quad (19)$$

Each local rule is equivalent to a base model in stacking.

C. Discussions and Future Research

Traditional stacking assigns each base model a constant weight. As pointed out in the previous subsection, a TSK fuzzy system can be viewed as an adaptive stacking model, because the weights for the base models (rule consequents) change with the inputs. Inspired by this phenomenon, we can design more powerful stacking strategies, by replacing each constant weight by a linear or nonlinear function[7] of the input $\mathbf{x}$. The rationale is that the weight for a base model should be dependent on its performance, whereas its performance is usually related to the location of the input: each base model may perform well in some input regions, but not well in the rest. A well-trained function of $\mathbf{x}$ may be able to reflect the expertise of the corresponding base model, and hence help achieve better aggregation performance.

Moreover, if the weighting functions and the base models are trained simultaneously, then the weighting functions may encourage the base models to cooperate: each focuses on a partition of the input domain, instead of the entire domain in traditional stacking. Even better performance could be expected, than training the base models first and then separately the weighting functions to aggregate them.

Some proven strategies in stacking may also be used to improve the performance of a TSK fuzzy system. For example, regularization is frequently used to increase the generalization of the stacking model [68], [74]. When LASSO [70] is used to build the stacking model, L_1 regularization is added, and hence some regression coefficients may be zero, i.e., it increases the sparsity of the solution. When ridge regression [71] or support vector regression [52] is used to build the stacking model, L_2 regularization is added, and hence the regression coefficients usually have small magnitudes, i.e., they reduce overfitting. Some new regularization terms, e.g., negative correlation [74], can be used to create negatively correlated base models to encourage specialization and cooperation among them. These concepts may also be used in training the rule consequents (base models) of a TSK fuzzy system, and also the antecedent MFs (so that the MFs for the same input are neither too crowded, nor too far away from each other).

VI. CONCLUSION

TSK fuzzy systems have achieved great success in numerous applications. However, there are still many challenges in designing an optimal TSK fuzzy system, e.g., how to efficiently optimize its parameters, how to balance the trade-off between

cooperations and competitions among the rules, how to overcome the curse of dimensionality, how to improve its performance without adding too many parameters, etc. Literature has shown that by making appropriate connections between fuzzy systems and other machine learning approaches, good practices from other domains may be used to improve the fuzzy systems, and vice versa.

This article has given an overview on the functional equivalence between TSK fuzzy systems and four classic machine learning approaches—neural networks, MoE, CART, and stacking—for regression problems. We also pointed out some promising new research directions, inspired by the functional equivalence, that could lead to solutions to the aforementioned problems. For example, by making use of the functional equivalence between TSK fuzzy systems and some neural networks, we can design more efficient training algorithms for TSK fuzzy systems; by making use of the functional equivalence between TSK fuzzy systems and MoE, we may be able to achieve a better trade-off between cooperations and competitions of the rules in a TSK fuzzy system; by making use of the functional equivalence between TSK fuzzy systems and CART, we can better initialize a TSK fuzzy system to deal with the curse of dimensionality; and, inspired by the connections between TSK fuzzy systems and stacking, we may design better stacking models, and increase the generalization of a TSK fuzzy model.

To our knowledge, this article is so far the most comprehensive overview on the connections between fuzzy systems and other popular machine learning approaches, and hopefully will stimulate more hybridization between different machine learning algorithms.

REFERENCES

[1] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions*, 2nd ed., Cham, Switzerland: Springer, 2017.
[2] C.-T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Upper Saddle River, NJ, USA: Prentice Hall, 1996.
[3] S. S. L. Chang and L. A. Zadeh, "On fuzzy mapping and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, no. 1, pp. 30–34, Jan. 1972.
[4] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan./Feb. 1985.
[5] J. Buckley, "Sugeno type controllers are universal controllers," *Fuzzy Sets Syst.*, vol. 53, pp. 299–303, 1993.
[6] L.-X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 807–814, Sep. 1992.
[7] D. Wu and J. M. Mendel, "Recommendations on designing practical interval type-2 fuzzy systems," *Eng. Appl. Artif. Intell.*, vol. 95, pp. 182–193, 2019.
[8] D. Wu and W. W. Tan, "Genetic learning and performance evaluation of type-2 fuzzy logic controllers," *Eng. Appl. Artif. Intell.*, vol. 19, no. 8, pp. 829–841, 2006.
[9] L.-X. Wang and J. M. Mendel, "Back-propagation of fuzzy systems as nonlinear dynamic system identifiers," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, San Diego, CA, USA, 1992, pp. 1409–1418.
[10] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[11] R. R. Yager and D. P. Filev, "Generation of fuzzy rules by mountain clustering," *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 209–219, 1994.
[12] M. Delgado, A. F. Gómez-Skarmeta, and F. Martín, "A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 223–233, May 1997.

---

[7]This idea was first used in [73] for *classification*, under the name "modified stacked generalization." It outperformed traditional stacking.

[13] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, no. 3, pp. 267–278, 1994.

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

[15] L. Jing, M. K. Ng, and J. Z. Huang, "An entropy weighting *k*-means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1026–1041, Aug. 2007.

[16] H. Jia and Y.-M. Cheung, "Subspace clustering of categorical and numerical data with an unknown number of clusters," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3308–3325, Aug. 2018.

[17] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[18] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford University Press, 1995.

[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 1991.

[20] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*, 1st ed., Boca Raton, FL: Routledge, 2017.

[21] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca-Raton, FL, USA: CRC Press, 2012.

[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[23] S. K. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world applications," *Fuzzy Sets Syst.*, vol. 65, no. 1, pp. 1–12, 1994.

[24] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 724–740, Sep. 1992.

[25] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets Syst.*, vol. 66, no. 1, pp. 1–13, 1994.

[26] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.

[27] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.

[28] R. Murray-Smith and T. A. Johansen, "Local learning in local model networks," in *Proc. 4th IEEE Int. Conf. Artif. Neural Netw.*, Perth, Australia, Jun. 1995, pp. 40–46.

[29] J.-S. R. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 156–159, Jan. 1993.

[30] K. J. Hunt, R. Haas, and R. Murray-Smith, "Extending the functional equivalence of radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 776–781, May 1996.

[31] J. Yen, L. Wang, and C. W. Gillespie, "Improving the interpretability of TSK fuzzy models by combining global learning and local learning," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 4, pp. 530–537, Nov. 1998.

[32] G. Golub, V. Klema, and G. W. Stewart, "Rank degeneracy and least squares problems," Tech. Rep. STAN-CS-76-559, Department of Computer Science, Stanford University, 1976.

[33] A. O. Hoori and Y. Motai, "Multicolumn RBF network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 766–778, Apr. 2018.

[34] C. Chen, R. John, J. Twycross, and J. M. Garibaldi, "An extended ANFIS architecture and its learning properties for type-1 and interval type-2 models," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Vancouver, Canada, Jul. 2016, pp. 602–609.

[35] C. Chen, R. John, J. Twycross, and J. M. Garibaldi, "Type-1 and interval type-2 ANFIS: A comparison," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Naples, Italy, Jul. 2017, pp. 1–6.

[36] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: Overview and comparisons," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 80–99, Feb. 2013.

[37] D. Wu and W. W. Tan, "Computationally efficient type-reduction strategies for a type-2 fuzzy logic controller," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Reno, NV, May 2005, pp. 353–358.

[38] N. Srivastava, G. Hinton, 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.

[39] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, Jun. 2013, pp. 1058–1066.

[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 448–456.

[41] D. Wu, Y. Yuan, and Y. Tan, "Optimize TSK fuzzy systems for big data regression problems: Mini-batch gradient descent with regularization, DropRule and AdaBound (MBGD-RDA)," 2019, *arxiv preprint*. [Online]. Available: https://arxiv.org/abs/1903.10951

[42] Y. Cui and D. Wu, "Optimize TSK fuzzy systems for big data classification problems: Bag of tricks," *IEEE Trans. Fuzzy Syst.*, submitted for publication.

[43] K. Amarasinghe and M. Manic, "Explaining what a neural network has learned: Toward transparent classification," in *Proc. Int. Joint Conf. Neural Netw.*, Rio, Brazil, Jul. 2018.

[44] D. Wu and J. M. Mendel, "Linguistic summarization using IF-THEN rules and interval type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 1, pp. 136–151, Feb. 2011.

[45] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 4, pp. 1006–1012, Aug. 2017.

[46] Y.-J. Zheng, W.-G. Sheng, X.-M. Sun, and S.-Y. Chen, "Airline passenger profiling based on fuzzy deep machine learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 12, pp. 2911–2923, Dec. 2017.

[47] H. D. Nguyen, L. R. Lloyd-Jones, and G. J. McLachlan, "A universal approximation theorem for mixture-of-experts models," *Neural Comput.*, vol. 28, no. 12, pp. 2585–2593, 2016.

[48] H. Bersini and G. Bontempi, "Now comes the time to defuzzify neuro-fuzzy models," *Fuzzy Sets Syst.*, vol. 90, no. 2, pp. 161–169, 1997.

[49] H. Andersen, A. Lotfi, and L. Westphal, "Comments on 'Functional equivalence between radial basis function networks and fuzzy inference systems' [with reply]," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1529–1532, Nov. 1998.

[50] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1177–1193, Aug. 2012.

[51] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: A literature survey," *Artif. Intell. Rev.*, vol. 42, no. 2, pp. 275–293, 2014.

[52] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

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

[54] M. Komijani, C. Lucas, B. N. Araabi, and A. Kalhor, "Introducing evolving Takagi-Sugeno method based on local least squares support vector machine models," *Evolving Syst.*, vol. 3, no. 2, pp. 81–93, 2012.

[55] J. Leski, "A fuzzy if-then rule-based nonlinear classifier," *Int. J. Appl. Math. Comput. Sci.*, vol. 13, pp. 215–223, 2003.

[56] M. W. Berry and M. Browne, Eds., *Lecture Notes in Data Mining*. Singapore: World Scientific, 2006.

[57] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 5, pp. 5–32, 2001.

[58] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[59] W.-Y. Loh, "Fifty years of classification and regression trees," *Int. Statistical Rev.*, vol. 82, no. 3, pp. 329–348, 2014.

[60] W. P. Alexander and S. D. Grimshaw, "Treed regression," *J. Comput. Graphical Statist.*, vol. 5, no. 2, pp. 156–175, 1996.

[61] P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao, "Piecewise-polynomial regression trees," *Statistica Sinica*, vol. 4, no. 1, pp. 143–167, 1994.

[62] J. R. Quinlan, "Learning with continuous classes," in *Proc. 5th Australian Joint Conf. Artif. Intell.*, Hobart, Tasmania, Nov. 1992, vol. 92, pp. 343–348.

[63] A. Dobra and J. Gehrke, "SECRET: A scalable linear regression tree algorithm," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Edmonton, Canada, Jul. 2002, pp. 481–487.

[64] R. L. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, no. 1, pp. 28–35, Jan. 1977.

[65] J.-S. R. Jang, "Structure determination in fuzzy modeling: A fuzzy CART approach," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Orlando, FL, USA, Jun. 1994, pp. 480–485.

[66] C. Z. Janikow, "Fuzzy decision trees: Issues and methods," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 28, no. 1, pp. 1–14, Feb. 1998.

[67] A. Suárez and J. F. Lutsko, "Globally optimal fuzzy decision trees for classification and regression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 12, pp. 1297–1311, Dec. 1999.

[68] D. Wu, F. Liu, and C. Liu, "Active stacking for heart rate estimation," *Inf. Sci.*, submitted for publication.

[69] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. New York, NY, USA: Chapman & Hall, 1993.

[70] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statistical Soc.*, vol. 58, no. 1, pp. 267–288, 1996.

[71] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[72] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," *Fuzzy Sets Syst.*, vol. 86, no. 3, pp. 251–270, 1997.

[73] R. Ebrahimpour, H. Nikoo, S. Masoudnia, M. R. Yousefi, and M. S. Ghaemi, "Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange," *Int. J. Forecasting*, vol. 27, no. 3, pp. 804–816, 2011.

[74] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Netw.*, vol. 12, no. 10, pp. 1399–1404, 1999.

Dongrui Wu (S'05–M'09–SM'14) received the B.E. degree in automatic control from the University of Science and Technology of China, Hefei, China, in 2003, the M.E. degree in electrical engineering from the National University of Singapore, Singapore, in 2005, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2009.

He is currently a Professor with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, and Deputy Director of the Key Laboratory of Image Processing and Intelligent Control, Ministry of Education, China. He has more than 120 publications, including a book entitled *Perceptual Computing* (Wiley-IEEE Press, 2010). His research interests include affective computing, brain–computer interface, computational intelligence, and machine learning.

Dr. Wu received the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award in 2012, the IEEE TRANSACTIONS ON FUZZY SYSTEMS Outstanding Paper Award in 2014, the North American Fuzzy Information Processing Society (NAFIPS) Early Career Award in 2014, the IEEE Systems, Man, and Cybernetics (SMC) Society Early Career Award in 2017, and the IEEE SMC Society Best Associate Editor Award in 2018. He was also a finalist of another three Best Paper Awards. He was/is an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS (2011–2018), IEEE TRANSACTIONS ON HUMAN–MACHINE SYSTEMS (2014), IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE (2017), and IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING (2019).
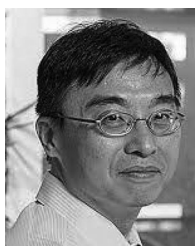
Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively.

He is currently a Distinguished Professor with the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, Australia, the University Chair Professor of electrical and computer engineering with National Chiao Tung University, Hsinchu, Taiwan, the International Faculty of the University of California at San Diego, San Diego, CA, USA, and the Honorary Professorship of the University of Nottingham, Nottingham, U.K. He has authored over 200 journal papers including approximately 110 IEEE journal papers in the areas of fuzzy systems, neural networks, and cognitive neuro-engineering.

Dr. Lin was elevated to an International Fuzzy Systems Association Fellow in 2012. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS from 2011 to 2016. He also served on the Board of Governors at the IEEE Circuits and Systems Society from 2005 to 2008, IEEE Systems, Man, and Cybernetics Society from 2003 to 2005, and IEEE Computational Intelligence Society (CIS) from 2008 to 2010. He is a Distinguished Lecturer of the IEEE CIS Society from 2015 to 2017. He served as the Deputy Editor-in-Chief for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II from 2006 to 2008.

Jian Huang (M'07–SM'17) received the B.E., M.Eng., and Ph.D. degrees from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1997, 2000, and 2005, respectively.

From 2006 to 2008, he was a Postdoctoral Researcher with the Department of Micro-Nano System Engineering and Department of Mechano-Informatics and Systems, Nagoya University, Nagoya, Japan. In 2015, he was a Research Fellow with Nagoya University, supported by JSPS invitation fellowship. He is currently a Full Professor with the School of Artificial Intelligence and Automation, HUST, Wuhan, China. He is also a Guest Professor with the Nagoya University of Japan and University Paris-Est Creteil (UPEC) of France. He has 14 authorized patents. He has published more than 190 papers (including 22 research articles in several IEEE Transactions and more than 50 conference papers in many IEEE conferences). His main research interests include rehabilitation robot, robotic assembly, networked control systems, and bioinformatics.

Dr. Huang serves as an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, Editor of the Springer *ROBOMECH Journal*, and Technical Editor of *PLoS ONE*. He has received the Grand Prize of Science and Technology award of China General Chamber of Commerce and golden medal at Geneva Inventions in 2017.

Zhigang Zeng (SM'07) received the Ph.D. degree in systems analysis and integration from the Huazhong University of Science and Technology, Wuhan, China, in 2003.

He is currently Professor and Dean of the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, and Director of the Key Laboratory of Image Processing and Intelligent Control, Education Ministry of China, Wuhan, China. He has published over 100 international journal papers. His current research interests include theory of functional differential equations and differential equations with discontinuous right-hand sides, and their applications to dynamics of neural networks, memristive systems, and control systems.

Dr. Zeng was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2010 to 2011, and has been an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS since 2014, and IEEE TRANSACTIONS ON FUZZY SYSTEMS since 2016. He has been an Editorial Board Member of *Neural Networks* since 2012, *Cognitive Computation* since 2010, and *Applied Soft Computing* since 2013.