Topology Learning-Based Fuzzy Random Neural Networks for Streaming Data Regression

Hang Yu, Jie Lu, *Fellow, IEEE*, and Guangquan Zhang

***Abstract*—As a type of evolving-fuzzy system, the evolving-fuzzy-neuro (EFN) system uses the structure inspired by neural networks to determine its parameters (fuzzy sets and fuzzy rules), so EFN system can inherit the advantages of neural networks. However, for streaming data regression, EFN systems still have several drawbacks: determining fuzzy sets is not robust to data sequence; determining fuzzy rules is complex as subspaces that can approximate to a Takagi–Sugeno–Kang (TSK) rule need to be obtained, and many parameters need to be optimized; and it is difficult to detect and adapt to changes in the data distribution, i.e., concept drift, if the output is a continuous variable. Hence, in this article, a novel evolving-fuzzy-neuro system, called the topology learning-based fuzzy random neural network (TLFRNN), is proposed. In TLFRNN, an online topology learning algorithm is designed to self-organize each layer of TLFRNN. Different from current EFN systems, TLFRNN learns multiple fuzzy sets to reduce the impact of noises on each fuzzy set, and a randomness layer is designed, which assigning the probability of each fuzzy set. Also, TLFRNN does not utilize TSK rules; instead uses a simple inference that considers fuzzy and random information of data simultaneously. More importantly, in TLFRNN, concept drift can be detected and adapted easily and rapidly. The experiments demonstrate that TLFRNN achieves superior performance compared to other EFSs.**

***Index Terms*—Concept drift, fuzzy random, neural networks, streaming data.**

I. Introduction

THE PROBLEM of streaming data mining has been a topic of consistent research in the fuzzy systems community [1], [2] because the generation of streaming data commonly occurs in an uncertain environment. In order to obtain knowledge from streaming data, a fuzzy system needs to meet four learning requirements: recursive and fast learning: data processing is noniterative; incremental or online learning: data are presented and learnt one at a time [3]–[4]; memory-efficient learning: no need to save previously processed data; and adaptive learning: the structure and parameters of a model can be changed to adapt to concept drift [5], [6]. To address these issues, evolving fuzzy systems (EFSs) [7] have been proposed.

Manuscript received July 2, 2020; revised September 27, 2020 and November 8, 2020; accepted November 16, 2020. Date of publication November 20, 2020; date of current version February 3, 2022. This work was supported by the Australian Research Council under Grant DP 190101733. *(Corresponding author: Jie Lu.)*

The authors are with the Decision Systems and e-Service Intelligence Laboratory, Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: hang.yu@student.uts.edu.au; jie.lu@uts.edu.au; guangquan.zhang@uts.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TFUZZ.2020.3039681.

Digital Object Identifier 10.1109/TFUZZ.2020.3039681

EFSs can learn incrementally, possibly even in real-time, on streaming data, and they are also able to adapt to changes in environmental conditions or properties of the data generation process [8]. Normally, EFSs are designed based on fuzzy rules [9]–[11] or fuzzy-neuro systems [13]–[15]. Compared to evolving-fuzzy-rule systems, evolving-fuzzy-neuro (EFN) systems use structure inspired by neural networks [20] to determine their parameters in fuzzy sets and fuzzy rules, so it inherits the advantages of neural networks. In most EFN systems, an online clustering method is first used to self-organize the structure of the fuzzy-neuro system (represented by a neural network). The parameters of the fuzzy-neuro system are learned by a backpropagation method. However, compared with other mining tasks, EFN systems face the following challenges in relation to streaming data regression: a more accurate system structure is needed; more parameters need to be optimized; and it is more difficult to detect and adapt to changes in the data distribution, i.e., concept drift.

Concept drift [18]–[19] is an essential challenge of streaming data mining. In streaming data, concept drift refers to data distribution $D_t(X,\ y) \neq D_{t+1}(X, y)$ at timestamp $t+1$, to lead to $f_t(X) \neq f_{t+1}(X)$. Current EFSs not robust in relation to detecting the concept drift of the streaming data, especially in regression problems. The reason for this difficulty is that the data distribution $D_t(X,\ y)$ is difficult to learn when y is a continuous variable. Hence, many EFSs transform the problem of detecting concept drift to the problem of detecting a change in one proposed variable based on fuzzy rules. For example, in relation to fuzzy rules, Lughofer [47] proposed a variable called "age" and Pratama [48] proposed relative mutual information. In [61], the error rate of the prediction of fuzzy rules is used to detect concept drift. Furthermore, as for these different variables, different conditions are proposed to identify whether the changed degree of these variables is too small to trigger a concept drift warning. However, these methods are designed based on the assumption that data distribution must follow some type of distribution. For example, in [47], the data distribution is assumed to follow the Gaussian distribution. In practice, the data distribution is complex and unknown. Recently, several EFSs [1], [12], such as the multilayer ensemble evolving fuzzy inference system [12], which make no assumptions on the data distribution model were proposed to overcome this drawback.

In summary, the performance of current EFSs for streaming data regression is still limited. Hence, a novel EFN system, called the topology learning-based fuzzy random neural network (TLFRNN), is proposed. In TLFRNN, similar to current EFN systems, we still use an online clustering method to learn a

neural network as the structure of the system and utilize fuzzy logic [21]–[22] to make an inference. However, the experimental results show that our proposed TLFRNN performs better than other EFSs methods. The major contributions of this article are as follows.

1) It proposes a new online clustering method that inherits the idea of topology learning [23]–[24]. In this method, the fuzzy set layer can be self-organized to a topology network after a single pass scan of the training dataset. Based on the topology network, the learning process of the fuzzy set layer robust to noises, and the topology network can accurately represent underlying data distribution.
2) It proposes a new type of structure for the EFN system based on the topology network. In this new type of structure, multiple fuzzy sets [25] can be included, and each fuzzy set is assigned to a probability by a randomness neuro. As a result, even some fuzzy sets have not been properly determined; the prediction accuracy of the system is still guaranteed.
3) It proposes an online mechanism to deal with the concept drift based on the topology network. Unlike current EFSs, our mechanism directly utilize $P_t(X,\ y)$ to detect concept drift, and thereby making TLFRNN detect and adapt to concept drift rapidly and easily.
4) It proposes a new type of fuzzy rules based on the topology network. In TLFRNN, determining fuzzy rules does not need to split data spaces of input-output variables to obtain the subspaces that can approximate to a Takagi–Sugeno–Kang (TSK) rule. As a result, only one parameter needs to be optimized, and this parameter can be automatically optimized by a maximum likelihood process.

The rest of this article is structured as follows. Section II introduces related works. Section III describes how to learn the structure of TLFRNN based on the topology network. Next, Section IV details how to determine the parameters of TLFRNN. Section V presents the experiment results. Finally, Section VI concludes this article.

II. Related Works

In this section, we introduce some research works related to TLFRNN.

A. EFN Systems

As a type of EFSs [26]–[28], to handle regression, the rules of EFN systems are mainly the first-order TSK type [29]–[30]

$$\begin{aligned}&\mathrm{Rule}_i : \mathrm{IF}\ \left(x_1 is A_1^i\right)\ \mathrm{AND}\ \ldots\ \mathrm{AND}\ \left(x_n\, is\, A_n^i\right)\\&\mathrm{THEN}\ \left(y_i = a_0^i + a_1^i x_1 + \cdots + a_n^i x_n\right)\end{aligned} \tag{1}$$

where Rule_i $(i = 1, 2, \ldots, R)$ is the *i*th TSK rule [31], x_j $(j = 1, 2, \ldots, n)$ is the *j*th input variable, and A_j^i is the *j*th antecedent of the *i*th rule. y_i $(i = 1, 2, \ldots, R)$ is the output variable of the *i*th rule, and a_j^i $(\mathrm{j} = 0, 1, 2, \ldots, n)$ is the *j*th coefficients in the consequent part of the *i*th rule.

When a TSK system receives input data X, each $Rule_i$ in the TSK system will be fired with a firing strength μ^i [32]–[33].

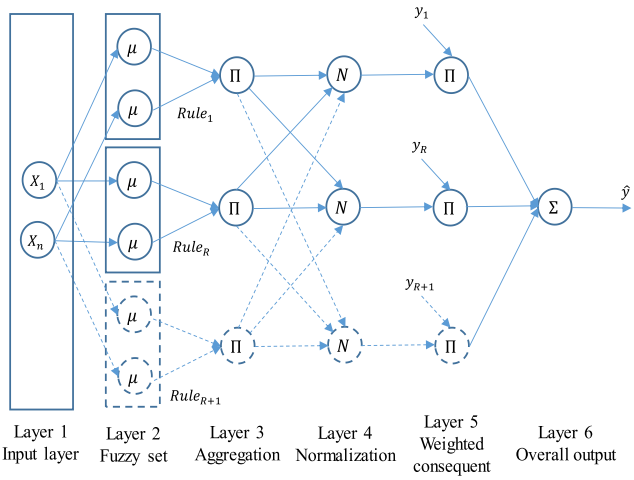

Fig. 1. EFN systems.

Correspondingly, the final estimated output of the system $(\hat{y})$ is defined in terms of a weighted average of the output produced by each Rule_i

$$\mu^i = \prod_{j=1}^{n} \mu_j^i\left(x_j\right) \text{ and } \hat{y} = \frac{\sum_{i=1}^{R} \mu^i y_i}{\sum_{i=1}^{R} \mu^i} \tag{2}$$

where the firing strength μ^i is the weight of the *i*th TSK rule, μ_j^i is the membership degree of the *j*th input variable, and $\hat{y}$ is the overall TSK system output.

An EFN system is essentially a neural network [34]–[35] that has the ability to adapt its structure and parameters to new input data through incremental/online learning. An illustration of an EFN system is shown in Fig. 1. The dotted neuros in the second layer represent new neuros that were added through learning.

The major characteristic of EFN systems is that the structure, i.e., fuzzy neural network, is self-constructed using a clustering method. According to the fuzzy neural network, the TSK rules can be determined [62]. Although some speedup methods for implementing fuzzy neural networks are proposed [63], self-constructing a fuzzy neural network is still a time-consuming task due to clustering [36]–[37], which is applied in the data space of input–output variables, and the resulting clusters are projected onto the input variables' coordinates. Usually, the number of neuros in the fuzzy set layer determines the number of rules because each of the neuros delineates a fuzzy region in the data space, meaning each of the fuzzy neuros defines a TSK rule. The coefficients in the consequent part of the TSK rule are learned separately using linear-optimized methods, such as RLS [38] and online gradient descent. However, different cost functions such as maximizing discriminability [64] and minimizing soft margin [65] have also been proposed for online learning parameters.

So far, many types of EFN systems have been proposed. For example, evolving fuzzy neural networks [39], sequential adaptive fuzzy inference systems [40], self-organizing fuzzy neural networks [41], and the dynamic evolving neural-fuzzy

inference system (DENFIS) [17] have been proposed. Furthermore, some EFNs are based on type-2 fuzzy sets, such as a self-evolving interval type-2 fuzzy neural network [42] and a TSK-type interval type-2 neural fuzzy system [66] have been proposed.

B. Fuzzy Random Variables

Fuzzy random variables were introduced by Kwakernaak [43] in 1978 to study randomness and fuzziness simultaneously. It was defined as a measurable function from a probability space to a collection of fuzzy numbers. For example, in a factory, the lifetime of some kinds of elements may be described as follows: "about five months" with probability 0.2, "about three months" with probability 0.4, and "about two months" with probability 0.4, where "about five months," "about three months" and "about two months" are all linguistics which can be characterized by fuzzy numbers or fuzzy variables.

Given a universe Γ, let *Pos* be a possibility measure that is defined for the power set $P(\Gamma)$ in Γ. Let $\Re$ be a set of real numbers. A function $Y : \Gamma \to \Re$ is a fuzzy variable defined as $\mu_Y(t) = Pos\{Y = t\}$. $t \in \Re$ is the possibility of event $\{Y = t\}$. Possibility Pos, necessity Nec, and credibility Cr of the event $\{Y \leq r\}$ for fuzzy variable *Y*, with possibility distribution μ_Y are

$$\begin{aligned} \mathrm{Pos}\{Y \leq r\} &= \sup_{t \leq r} \mu_Y(t) \\ \mathrm{Nec}\{Y \leq r\} &= 1 - \sup_{t > r} \mu_Y(t) \\ \mathrm{Cr}\{Y \leq r\} &= \frac{1}{2}\left(1 + \sup_{t \leq r} \mu_Y(t) - \sup_{t > r} \mu_Y(t)\right) \end{aligned} \tag{3}$$

where sup ("supremum") means, basically, the largest.

III. TLFRNN: Learning Structure

This section describes how to learn the structure of TLFRNN. Furthermore, the mechanism of TLFRNN handling concept drift is detailed in this section.

A. Learning Topology Network

In EFN systems, each element of input data *X* is mapped to a membership value between 0 and 1 in the fuzzy set layer. This membership value μ is later brought into the inference process and significantly affects the accuracy of prediction. The fuzzy set layer is self-organized by an online clustering method. However, the result of online clustering is sensitive to the initial clusters. With an inappropriate selection of initial clusters, the clustering results will be poor regardless of how the algorithm is adjusted later [44], thereby impacting the accuracy of the structure of the fuzzy set layer. Unfortunately, in streaming data, we are not able to determine which initial cluster is the best choice since we do not have information streaming data. To overcome this problem, we integrate the idea of fuzzy random variables [45] into the TLFRNN. Since the membership value is dependent on the initial cluster and the initial cluster is random if the training data are streaming data, the proposed TLFRNN will build multiple fuzzy sets, and the input data will be transferred into fuzzy random variables. Since the input data in the proposed TLFRNN will be converted into fuzzy random variables, TLFRNN has an extra layer called the randomness layer compared with the traditional structure of EFN systems (see Fig. 1). When a sample $Z(t) = (X(t), y(t))$, where $X(t) \in R^d$, and $y \in R^1$, arrives, according to the neuros in the randomness layer (called randomness neuro), we first get the probability P_i of this article. Then, according to the neuros in the fuzzy set layer (called fuzzy neuro), we can get fuzzy memberships μ_i of this sample.

Next, we need to propose an algorithm to self-organize the structure of our proposed TLFRNN. Self-organized topology learning [46] can learn a topology network online and has been shown to offer many advantages. For example, the learning process is robust to noise based on the topology network, and the topology network can represent the real data distribution. Hence, self-organized topology learning is a good choice for learning our proposed TLFRNN. However, two parameters need to be set manually in self-organized topology learning: λ is used to define the frequency of neuro removal. A relatively large λ value contributes to the deletion of fewer neuros. This implies that more neuros will remain. $\mathrm{age}_{\max}$ is defined as the lifetime of each edge between neuros. A relatively large $\mathrm{age}_{\max}$ value will result in a neuro having more neighbors. However, noise becomes hard to delete. The topology network is very sensitive to λ and $\mathrm{age}_{\max}$ parameters in the learning process. For example, a topology network that can accurately represent the real data distribution can be obtained only when $\lambda = 100$ and $\mathrm{age}_{\max} = 50$. Hence, we improved the self-organized topology learning to decrease the sensitivity of these two parameters.

Assume the fuzzy neuros will sustain two distinct topology networks, and each topology network needs to connect to a randomness neuro. Hence, assume streaming data $\{Z\}$ with samples $\{Z(1), \ldots, Z(t)\} \in R^{d+1}$, the process of learning neuros in the randomness layer and the fuzzy set layer is as follows.

In TLFRNN, at timestamp t, each randomness neuro has a value $W_i(t)$ and when sample $Z(t)$ is passed to the randomness layer, each randomness neuro will get a probability $P_i(Z(t))$ to represent the probability that the sample $Z(t)$ belongs to this randomness neuro

$$P_i(Z(t)) = \sum_{i=1}^{K} \omega_i(t) f(Z(t), W_i(t), \Sigma_i(t)) \tag{4}$$

where *K* is the number of neuros in the fuzzy set layer that connects this neuro, $\omega_i(t)$ is the weight of the *i*th component, such that $\sum_{i=1}^{K} \omega_i(t) = 1$, and $f(X(t), W_i(t), \Sigma_i(t))$is the Gaussian density distribution with mean $W_i(t)$ and covariance matrix $\Sigma_i(t)$. In this case, $\Sigma_i(t)$ is a diagonal matrix for *d* dimensional, as it is shown as

$$\Sigma_i(t) = \mathrm{diag}\left(\sigma_{i,1}^2, \sigma_{i,2}^2, \ldots, \sigma_{i,d}^2\right). \tag{5}$$

However, to reduce the interference of the parameter values of a neuro over the parameter values of other neuros. For this, we propose to normalize the means after multiplying them by the covariance matrix. Let $\Sigma_i^{-1}(t)$ be a positive semi-definite matrix and $\Sigma_i^{-1}(t) = AA^T$, furthermore, in our case $A = A^T$. Then,

we can write $f(X(t), W_i(t), \Sigma_i(t))$ as

$$f\left(X\left(t\right), W_i\left(t\right), \Sigma_i\left(t\right)\right) = \frac{1}{\sqrt{2\pi}\sigma_i^2\left(t\right)} \times \exp\left(\frac{X^{*T}\left(t\right) W_i^*\left(t\right) - 1}{2\sigma_i^2\left(t\right)}\right) \tag{6}$$

where $Z^*\left(t\right) = AZ(t), W_i^*\left(t\right) = AW_i(t)$.

Assume there are already two neuros in the randomness layer at timestamp t, and the first randomness neuro has value $W_1(t)$. At the next timestamp $t+1$. If sample $Z(t+1)$ is inserted into the topology network that connects the first randomness neuro, the $W_1^*(t+1)$ is updated according to

$$W_i^*\left(t+1\right) = W_i^*\left(t\right) + \left(K\omega_i\right)^{-1} p\left(1|Z\left(t\right)\right)\left(Z^*\left(t+1\right) - W_i^*\left(t\right)\right) \tag{7}$$

$$\sigma_i^2\left(t+1\right) = \sigma_i^2\left(t\right) + \left(K\omega_i\right)^{-1} p\left(i|Z\left(t\right)\right) \times \left[\left(Z^*\left(t+1\right) - W_i^*\left(t\right)\right)^2 - \sigma_i^2\left(t\right)\right] \tag{8}$$

$$\omega_i\left(t+1\right) = \omega_i\left(t\right) + K^{-1}\left(p\left(1|Z\left(t\right)\right) - \omega_i\left(t\right)\right) \tag{9}$$

where

$$p\left(i|Z\left(t\right)\right) = \frac{\omega_i f\left(Z\left(t\right), W_i\left(t\right), \Sigma_i\left(t\right)\right)}{P\left(Z\left(t\right)\right)}. \tag{10}$$

Next, we use fuzzy neuros that are connected to the first randomness neuro $W_1(t)$ as an example to illustrate learning a topology network. Each fuzzy neuro has a value $F_i(t)$, so we use $F_i(t)$ to refer to a fuzzy neuro in the following chapters. In TLFRNN, the learning objective of the topology network is defined as a minimization of the reconstruction error

$$\mathrm{Error}_{\mathrm{reconstruction}} = \sum_{\mathrm{t}=1}^{k} \sum_{i \in N} \mu_i\left(t\right) d_i^2\left(t\right) \tag{11}$$

where $d_i(t)$ defines the Euclidian distance (L_2-norm) between the sample $Z(t)$ and the fuzzy neuro $F_i(t)$ as follows:

$$d_i^2\left(t\right) = \left\|\left(Z\left(t\right) - F_i\left(t\right)\right)\right\|^2, 1 \le t \le k, 1 \le i \le N \tag{12}$$

and *N* is the set of neuros and

$$0 \le \mu_i\left(t\right) \le 1, \sum_{i \in N} \mu_i\left(t\right) = 1. \tag{13}$$

Different with other self-organized topology learning methods, the $\mu_i(t)$ of the fuzzy neuro F_i in our method is a fuzzy membership which is calculated as

$$\mu_i\left(t\right) = \left[d_i^2\left(t\right) \sum_{j=1}^{N} \left(\frac{1}{d_j^2\left(t\right)}\right)\right]^{-1} \tag{14}$$

where $\mu_i(t)$ can represent the degree to which the input data $Z(t)$ belongs to the fuzzy neuro $F_i(t)$, so a fuzzy neuro $F_i(t)$ with maximum fuzzy membership $\mu_i(t)$ means it is the nearest fuzzy neuro to the sample $Z(t)$.

The details of the learning topology network are shown in Algorithm. 1. From Algorithm. 1, we can see the first step is to determine whether or not to insert a new sample $Z(t)$ in the fuzzy set layer, i.e., as a new fuzzy neuro $F_i(t)$. Assume we already have N fuzzy neuros $\{F_1(t), \ldots, F_N(t)\}$ existing in the topology network at timestamp t. When a new sample $Z(t)$ is given, it finds the nearest fuzzy neuro (winner) $F_1'(t)$ and the second-nearest fuzzy neuro $F_2'(t)$ (s-winner) of the sample $Z(t)$ by the fuzzy membership $\mu_i(t)$. If the fuzzy membership $\mu_i(t)$ between the new sample $Z(t)$ and the winner $F_1'(t)$ or s-winner $F_2'(t)$ is more than the threshold T_i, and if no edge connects $F_1'(t)$ and $F_2'(t)$, $F_1'(t)$ and $F_2'(t)$ should be connected with an edge. The "age" of the edge is set as "0," and the age of all the edges linked to the $F_1'(t)$ is subsequently increased by "1." Then an edge is deleted if its age exceeds $\mathrm{age}_{\max}$. Otherwise, the new sample $Z(t)$ is inserted into the topology network as new fuzzy neuro $F_{N+1}(t)$. When the number of input samples equals λ, the deletion of fuzzy neuros will be triggered.

If a fuzzy neuro $F_i(t)$ has some fuzzy neuros that connect to it, the threshold T_i is calculated using the maximum fuzzy membership $\mu_i(t)$ of the connected neuros

$$T_i = \max_{i \in \mathrm{con}_i} \mu_i\left(t\right) \tag{15}$$

where con_i is a set of all fuzzy neuros that connect to the fuzzy neuro $F_i(t)$.

If a neuro $F_i(t)$ has no fuzzy neuros that connect to it, the threshold T_i is defined as the minimum fuzzy membership $\mu_i(t)$

Algorithm 1. Learning topology network

Input: Sequence $\{Z\}$, age_{max}, λ
Output: The topology network
1: Initialize set AN with the first 2 samples drawn from $\{Z\}$
2: while $\{Z\}$ is not empty **do**
3: Find winner F_1' and second winner F_2' from neuro set AN, as
$F_1' = F_i$ with $\max_{i \in AN} \mu_i$
$F_2' = F_i$ with $\max_{i \in AN \setminus \{F_1'\}} \mu_i$
4: **if** $\max_{i \in AN} \mu_i < T_1$ **and** $\max_{i \in AN \setminus \{F_1'\}} \mu_i < T_2$ **then**
5: Add a neuro F_i with weight $Z(t)$ to AN, and set active times $C_i = 0$.
6: Update W_1^*, σ_1^2, ω_1 according to Eq. (7)- Eq. (9)
7: **else**
8: Increase the active times as $C_{F_1'}(t) = C_{F_1'}(t-1) + 1$. Then update neuros in the fuzzy set layer as
$F_1'(t+1) = F_1'(t) + \gamma_i(t+1)\Delta F_1'(t+1)$
$F_i(t+1) = F_i(t) + \lambda \cdot \gamma_i(t+1) \cdot \Delta F_i'(t+1)$ for all neuros F_i that connect to F_1'
9: Increase the active time C_i of all neuros linked with F_1' by 1
10: Increase the age of all edges linked with F_1' by 1.
11: **for all** edge $age_{i,j} > age_{max}$ do
12: Remove the edge connecting F_i and F_j.
13: **end for**
14: **end if**
15: **if** number of input samples divides λ **then**
16: Remove neuros F_i that only one neuro connects to it.
17: Remove neuros F_i that the active time C_{F_i} does not increase.
18: **end if**
19: end while

of the other fuzzy neuros

$$T_i = \min_{i \in AN \setminus \{i\}} \mu_i(t) \tag{16}$$

where AN is a set of all fuzzy neuros that exists in a topology network.

The next step is to update the $F_1'(t)$ whereby all fuzzy neuros $F_i(t)$ that connect to $F_1'(t)$ are updated. Assume a new fuzzy neuro $F_i(1)$ is inserted into a topology network at timestamp 1. It follows that the value of the fuzzy neuro $F_i(t)$ at timestamp t is defined as follows:

$$F_{\mathrm{i}}(t) = \frac{\sum_{k=1}^{t} \mu_i^2(t) Z(t)}{\sum_{k=1}^{t} \mu_i^2(t)} \tag{17}$$

where $\mu_i(t)$ denotes the fuzzy membership of the data $Z(t)$.

However, if we use (17) to directly calculate the fuzzy neuro $F_i(t)$, we need to save $\mu_i(t)$ and $Z(t)$ during each learning iteration. This conflicts with the requirements of handling online streaming data. Therefore, we need to develop a new method to calculate the fuzzy neuro $F_i(t)$.

In online learning, the relationship between an old fuzzy neuro F_i and a new one is as follows:

$$F_i(t+1) = F_i(t) + \Delta F_i(t+1) \tag{18}$$

where the $\Delta F_i(t+1)$ is

$$\Delta F_i(t+1) = \frac{\mu_i^2(t+1)\left(Z(t+1) - F_i(t)\right)}{U_i(t+1)} \tag{19}$$

and the denominator $U_i(t+1) \in R^d$ is defined as

$$U_i(t+1) = U_i(t) + \mu_i^2(t+1) \tag{20}$$

where $U_i(t)$ is defined as follows:

$$U_i(t) = \sum_{k=1}^{t} \mu_i^2(k). \tag{21}$$

According to the idea of online learning, the winner $F_1'(t)$ is updated as follows:

$$F_1^{'}(t+1) = F_1^{'}(t) + \gamma_i(t+1)\Delta F_1^{'}(t+1) \tag{22}$$

where the parameter $\gamma_i(t), (0 \le \gamma_v \le 1)$ is the reciprocal of the winning times W_i of winner $F_1'(t+1)$.

As for the other neuros $F_i(t+1)$ that connect to the winner $F_1'(t+1)$, these are updated as follows:

$$F_i(t+1) = F_i(t) + \lambda \cdot \gamma_i(t+1) \cdot \Delta F_i(t+1) \tag{23}$$

where λ is the learning rate which was chosen to decrease the number of samples by $\lambda = 1/k_i$, with k_i the number of samples which connect to the winner. Based on (22) and (23), all neuros F_i can be updated in an online manner.

In summary, multiple fuzzy sets can be learnt in our TLFRNN, and one fuzzy set will be assigned a probability. So the proposed TLFRNN is not sensitive to the randomness of streaming data. Furthermore, our proposed topology network learning method is robust λ and $\mathrm{age}_{\max}$ parameters. Hence, this reduces the need for accuracy of user-set parameters. Besides, we do not need to use generalized rules [66] because the size of the area of the topology network is independent of whether the corresponding ellipse is nonaxis parallel and the non-axis parallel generalized rule needs more calculation.

B. Concept Drift

The most important problem in streaming data mining is the existence of concept drift. Concept drift can be defined as a change in the data distribution. Detecting and adapting to concept drift is an important part of EFSs since concept drift increases the error prediction rate.

Most current EFSs [47]–[48] for regression cannot detect concept drift by directly detecting changes in data distribution as estimating data distribution $D(X, y)$ is a difficult task in regression problems. However, in our proposed TLFRNN, the learned neuros will build a topology network that is able to represent data distribution $D_{0,t}(X, y)$ accurately, so the change of distribution will map to the change in the topology network, and then the new data distribution $D_{0,t+1}(X, y)$ can be adapted by updating the topology network if $D_{0,t}(X, y) \neq D_{0,t+1}(X, y)$. In TLFRNN, adapting to concept drift does not mean the drift has to be detected. Since there are multiple topology networks in TLFRNN, when a new sample is given, the topology network updates itself from a global and local viewpoint. From a global viewpoint (i.e., for all topology networks), there is a forgetting mechanism conducted in the case of concept drift. In step. 5 of Algorithm 1, a variable *C* is used to record the number of neuros to be activated. If a neuro is active, this means it is selected as the winner or s-winner, or it can connect to the winner by edges in this learning iteration. Hence, in one iteration, we label the neuros in each topology network with no changed variable C as outdated neuros and delete them from each topology network. From a local viewpoint (i.e., at each topology network), as the value F_i of neuros will be updated according to step. 8 of Algorithm 1, the topology network will finally adapt to the new data distribution. Furthermore, the neuros in the randomness layer are also updated according to the updating of the topology network to which it connects.

C. Computational Complexity

There are two main operations of learning the topology network. Suppose N is the size of the nodes already in the topology network. When a new sample $Z(t)$ is inputted, we need to find μ_1' and μ_2' at the first stage, so the computational complexity is $O(N)$. Next, we need to scan all the nodes again in the worst-case scenario to get T_1 and T_2, i.e., μ_1' and μ_2' have no neighbors, so the computational complexity increases to $O(N^2)$. In the next stage, we need to update the information of the topology network, but not all nodes need to be scanned. Hence, the computational complexity of the next stages is $O(1)$. In summary, the total computational complexity of processing an input sample is $O\left(N^2 + N + 1\right) = O(N^2)$.

IV. TLFRNN: Determining Parameters

In this section, we describe how to determine and use the maximum likelihood process to optimize the parameters of TLFRNN.

A. Fuzzy Set and Fuzzy Rules

For streaming data regression, the inference of current EFSs is commonly based on first-order Takagi–Sugeno fuzzy rules [30], that is, the consequent $y_i(t)$ of each fuzzy rule is modeled by a linear function (1). Hence, determining fuzzy rules need to split data spaces of input-output variables to obtain the subspaces that can approximate to a TSK rule. However, in practical applications, the prediction problem in each subspace is still a nonlinear model and cannot approximate a TSK rule. In addition, due to the first-order TSK rule is a linear function, the prediction model of current EFSs is actually a combination of multiple linear models. As a result, current EFSs are difficult to model the nonlinear relationship between input and output. Furthermore, to obtain the consequent $y_i(t)$, several parameters such as the parametersa_j^i in (1) need to be learned.

In the proposed TLFRNN, each neuro $F_i(t)$ in the fuzzy set layer is a fuzzy rule and has a consequent $y_i(t)$, and different with the first-order Takagi–Sugeno fuzzy rule, the consequent $y_i(t)$ of each fuzzy rule is calculated by

$$\begin{aligned} y_i(t) &= f(X(t) \mid \{F_i(t)\}) \\ &= \frac{K_{h(t)}\left(X(t) - F_i^X(t)\right) F_i^y(t)}{\sum_{j=1}^{n} K_{h(t)}\left(X(t) - F_i^X(t)\right)} \end{aligned} \tag{24}$$

and the $K_{h(t)}(X(t) - F_i^X(t))$ is defined as

$$\begin{aligned} K_{h(t)}\left(X(t) - F_i^X(t)\right) &= \frac{1}{\sqrt{(2\pi)^d |H(t)|}} \cdot \\ \times \exp\left(-\frac{1}{2}\left(X(t) - F_i^X(t)\right)^T H(t)^{-1}\left(X(t) - F_i^X(t)\right)\right) \end{aligned} \tag{25}$$

where d is the dimension of the input data, $H(t)$ is a diagonal matrix with squares of smooth parameter $h_k(t)$, and $|H(t)| = \prod_k h_k(t)$.

Due to the consequent $y_i(t)$ of each fuzzy rule is represented by a kernel function, so the data spaces of input-output variables do not need to be split to obtain the subspaces that can approximate to a linear model. Hence, based on this type of fuzzy rules, the inference of an input data $X(\mathrm{t})$ is as follows.

Step 1: We first calculate the probability P_i that the input data $X(t)$ attaches each neuro in randomness layer by (4).

Step 2: We then obtain the fuzzy membership μ_i whereby input data $X(t)$ attaches to each neuro F_i in the fuzzy set layer using (14) and normalizes the fuzzy membership μ_i to obtain the weights

$$\theta_i = \sum_i \frac{\mu_i}{\sum_j \mu_j}. \tag{26}$$

Step 3: Next, given the input data $X(t)$, we further obtain the consequent $y_i(t)$ of each neuro F_i using (24).

Step 4: Like other fuzzy systems, we also weight the $y_i(t)$ to obtain a weighted value $y^*(t)$ by

$$y^*(t) = \sum_i \theta_i y_i(t). \tag{27}$$

Step 5: Finally, we obtain the prediction value $\hat{y}(t)$

$$\hat{y}(t) = \sum_i \frac{P_i(t)\, y_i^*(t)}{\sum_j P_j(t)}. \tag{28}$$

Therefore, different from current EFSs, in the inference of our proposed TLFRNN, the random and fuzzy information of samples are used at the same time, and the consequent $y_i(t)$ of each fuzzy rule is modeled as a nonlinear function.

B. Selecting Parameter

According to (25), only the smooth parameter $H(t)$ influences forecasting. Some methods, such as RLS and online gradient descent [49], are used to select the smooth parameter automatically. However, these methods require that the neuros F_i in the fuzzy set layer cannot be changed in the next learning iteration. However, in our proposed TLFRNN, the neuros F_i may be $F_i(t) \neq F_i(t+1)$. Hence, we develop a new method for automatically setting the appropriate smooth parameter parameter $H(t)$.

In the proposed method, at first, the smooth parameter $H(t)$ is redefined as follows:

$$H(t) = \mathrm{diag}\left(s(t)\,\delta_1^2(t), \ldots, s(t)\,\delta_k^2(t), \ldots s(t)\,\delta_d^2(t)\right) \tag{29}$$

so (26) can also be rewritten as

$$\begin{aligned} K_{h(t)}(X(t) - F_i(t)) &= \frac{1}{\sqrt{(2\pi)^d (s(t))^d \prod_k \delta_k^2(t)}} \cdot \\ \times \exp\left(-\frac{1}{2s(t)}(X(t) - F_i(t))^T \Delta(t)\,(X(t) - F_i(t))\right) \end{aligned} \tag{30}$$

where $\Delta(t) = \mathrm{diag}(\delta_1^{-2}(t), \ldots, \delta_k^{-2}(t), \ldots, \delta_d^{-2}(t))$.

However, as there is a fuzzy neuro $F_i(t)$ may change as new input data $X(t)$ arrives, $s(t)$ needs to be recalculated when new $X(t)$ arrives. Given input data $X(t)$, we denote the *k*th dimension of neuro F_i as $F_i^k$and we can obtain the possibility $\hat{P}(X(t))$ of input data $X(t)$ under moment $s(t)$ as

$$\begin{aligned} \widehat{P}(X(t) \mid s^*(t)) &= \sum_i K_{h(t)}(X(t) - F_i(t) \mid s^*(t)) \mu_i(t) \\ &= \sum_i \lambda_i(t)\,(s^*(t))^d \exp\left(-0.5 (s^*(t))^2 \chi_i(t)\right) \mu_i(t) \end{aligned} \tag{31}$$

where

$$\lambda_i(t) = \frac{1}{\sqrt{(2\pi)^d \prod_k \delta_k^2(t)}} \tag{32}$$

$$s^*(t) = s(t)^{-0.5} \tag{33}$$

$$\chi_i(t) = \sum_k \left(X_k(t) - F_i^k(t)\right)^2 \delta_k^{-2}(t). \tag{34}$$

We assume the $\hat{P}(X(t))$ of this moment input data $X(t)$ is the maximum under this moment smooth parameter $s^*(t)$, i.e. $\hat{P}_{\max}(X(t)s^*(t))$. Thus, we can obtain $\hat{P}_{\max}(X(t)s^*(t))$ by maximizing the likelihood function $L(s^*(t)X(t))$. Then, the

likelihood function $L(s^*(t)X(t))$ can be transformed into

$$L\left(s^*\left(t\right)|X\left(t\right)\right)=\sum_i \mu_i\left(t\right)L\left(s^*\left(t\right)|X\left(t\right),F_i\left(t\right)\right). \quad (35)$$

Hence, maximizing $L(s^*(t)X(t))$ equates to maximizing $L(s^*(t)X(t),F_i(t))$ for each neuro $F_i(t)$. Next, we transform the likelihood function $L(s^*(t)X(t),F_i(t))$ into the log-likelihood function, so

$$L\left(s^*(t)|X\left(t\right)\right) = \sum_i \mu_i\left(t\right)\left(\ln\lambda_i\left(t\right)+d\ln s^*\left(t\right)-0.5(s^*\left(t\right))^2\chi_i\left(t\right)\right). \quad (36)$$

Let $\frac{\partial L(s^*(t)X(t))}{\partial s^*(t)}=0$, we have

$$s^*\left(t\right)=\sqrt{\frac{d\sum_i \mu_i\left(t\right)}{\sum_i \mu_i\left(t\right)\chi_i\left(t\right)}} \quad (37)$$

then according to $\sum_i \mu_i(t)=1$, we finally obtain

$$s^*\left(t\right)=\sqrt{\frac{d}{\sum_i \mu_i\left(t\right)\chi_i\left(t\right)}} \quad (38)$$

where $\sum_i \mu_i(t)\chi_i(t)\propto\sum_i \mu_i(t)\|X(t)-F_i(t)\|^2$.

According to the learning objection of the topology network, the number of neuros $F_i(t)$ in the fuzzy set layer increases gradually with the continuous arrival of the streaming data, and $\sum_i \mu_i(t)\|X(t)-F_i(t)\|^2$ decreases. Then due to $\sum_i \mu_i(t)\chi_i(t)\propto\sum_i \mu_i(t)\|X(t)-F_i(t)\|^2$, the decreasing of $\sum_i \mu_i(t)\|X(t)-F_i(t)\|^2$ will result in $s^*(t)$ decreasing. Next, according to (29) and (33), we can conclude that the smooth parameter $H(t)$ decreases as the number of neuros F_i increases in TLFRNN. As for parameter λ and $\mathrm{age}_{\max}$, choosing an appropriate value for each strongly depends on the nature of the data. In practice, if λ is set to over ½ of the size of the window, and $\mathrm{age}_{\max}$is set to over ¼ of λ, a satisfying prediction result can be obtained, but this is only a guess based on trial-and-error and is insensitive.

V. Experiments

In this section, we illustrate the advantages of the proposed TLFRNN by comparing its algorithm with other EFSs. As TLFRNN is an EFN system, we first validate our proposed algorithm for learning neural network. Then, we validate the regression performance of TLFRNN. The evaluations involve different scenarios using both artificial and real-world datasets. All experiments are conducted in Python 3.5 version on Windows 7 running on a PC with a system configuration Intel Core i5 processor (2.40 GHz) with 8-GB RAM.

In the experiment, we employed a prequential strategy which was introduced in [50]. In this strategy, a sample is first evaluated by a learning model and is then used to update the learning model. By considering all the metrics, we calculate their mean value for all data streams. In addition, as suggested by [51], a windowed measurement method is also considered, and a nonoverlapping sliding window of size 200 is also used. Aiming at reducing the effects of randomness in our evaluation, all multioutput regression methods are performed at least thirty times for each dataset, and the performance is taken as the average between the repetitions.

A. Learning Topology Network

To validate our proposed algorithm for learning topology network (for convenience, we call it TN in the following sections), we compare TN to fuzzy c-means (FCMs) [36] and SOINN [44]—selecting FCM to illustrate the advantage of topology learning and selecting SOINN to illustrate the advantage of fuzzy learning. The specific comparison scheme is: we first use FCM, SOINN, and TN to separately obtain the neural networks based on the following dataset:

$$\begin{cases} y\left(t\right)=\sin\left(o\left(t\right)\right)/20 \\ X\left(t\right)=o\left(t\right)/10-6.25 \end{cases} \quad (39)$$

where $o(t)$ is randomly sampled from (0, 5). To illustrate that our proposed method is also robust to noise, we add 10% noise to the dataset and the noise is distributed over the whole.

We then make a prediction of the given dataset based on the neural network (represented by neuros) obtained by each algorithm. By comparing the prediction results in terms of the average root-mean-square error (ARMSE), we can validate the performance of each algorithm.

Next, we discuss the parameters of each algorithm. When using FCM to obtain the neuros, it is necessary to select two parameters: C is used to define the number of neuros; and m is the fuzzifier m and determines the level of cluster fuzziness. A large m results in smaller membership values and, hence, fuzzier clusters. In the limit $m=1$, the memberships converge to 0 or 1, which implies a crisp partitioning. In the absence of experimentation or domain knowledge, m is commonly set to 2. TN does not need to define the number of neuros, but it needs to set λ and $\mathrm{age}_{\max}$. λ is used to define the frequency of neuro removal. $\mathrm{age}_{\max}$ is defined as the lifetime of each edge. SOINN also does not need to define the number of neuros, but it needs more parameters [44]. Except for the need to manually set λ and $\mathrm{age}_{\max}$, C_1 and C_2 also need to be set manually. The definition of λ and $\mathrm{age}_{\max}$ is the same as in our proposed algorithm.C_1 and C_2 are defined to control deletion behavior. A relatively large C_1 or C_2 value will contribute to higher noise tolerance; however, more useful neuros will be deleted at the same time.

Fig. 2 shows the topology network obtained by each algorithm using different parameters. Then, we set the smooth parameter to be equal to 0.5 to make a prediction. Table I lists the prediction results. From the first line of Table I, we can see that FCM achieves a good prediction result when the number of neuros is set to 20. However, when the number of neuros is set to 10, the prediction result declines, as shown in the second line of Table I. The reason is shown in Fig. 2(a) and (b). From Fig. 2(a), we can see the data distribution is well represented by neuros when their number is set to 30; but from Fig. 2(b), the data distribution

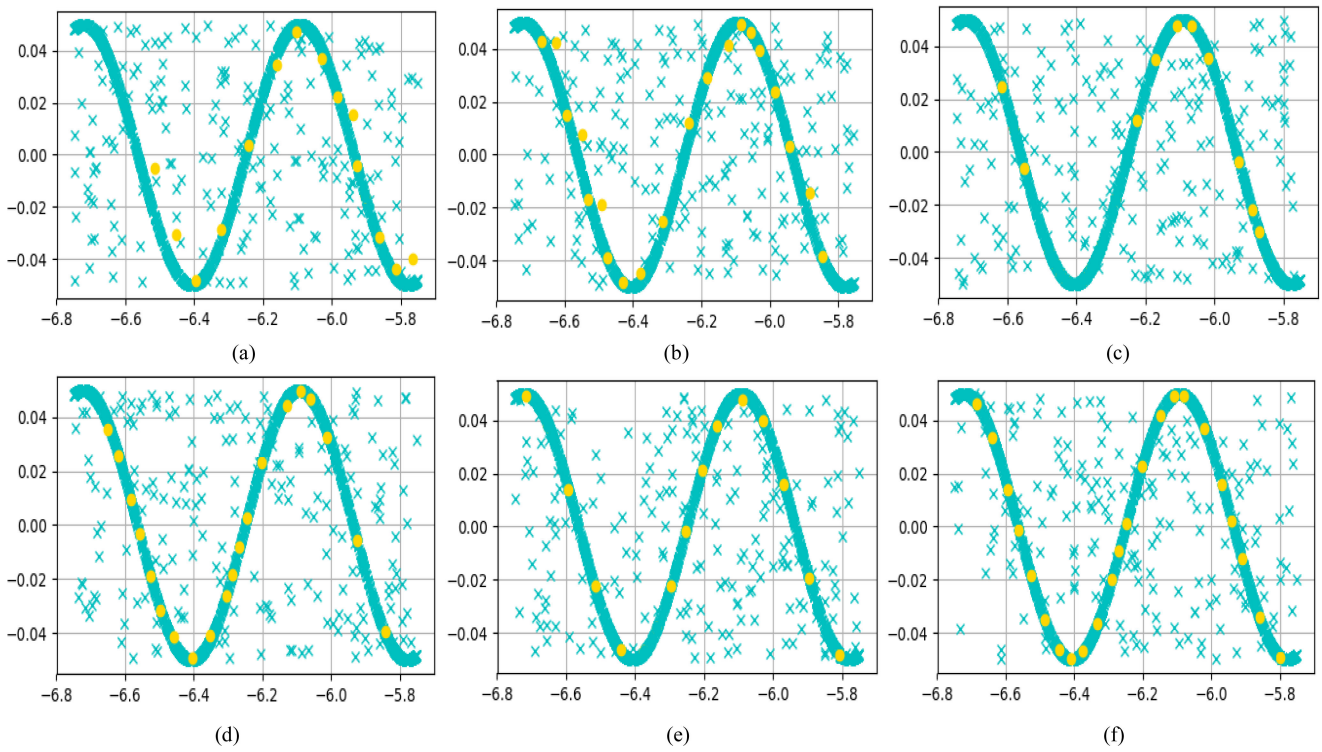

Fig. 2. Neuros obtained by each algorithm. (a) $C = 10$ and $m = 2$, FCM. (b) $C = 20$ and $m = 2$, FCM. (c) $\lambda = 100$ and $\text{age}_{\max} = 25$, SOINN. (d) $\lambda = 100$ and $\text{age}_{\max} = 50$, SOINN. (d) $\lambda = 100$ and $\text{age}_{\max} = 25$, TL. (e) $\lambda = 100$ and $\text{age}_{\max} = 50$, TL.

TABLE I
DIFFERENT SMOOTH PARAMETER SETTINGS

Clustering	Parameters	Neurons	ARMSE
FCM	C = 10 *and* $m = 2$,	10	0.004071
	C = 10 *and* $m = 2$	20	0.03652
SOINN	$\lambda = 100$ and $\text{age}_{\max} = 25$	13	0.003823
	$\lambda = 200$ and $\text{age}_{\max} = 50$	19	0.003712
TN	$\lambda = 100$ and $\text{age}_{\max} = 25$	13	0.003623
	$\lambda = 200$ and $\text{age}_{\max} = 50$	23	**0.003492**

cannot be well represented by neuros when the number is set to 10. Therefore, if we want to use FCM to obtain the neuros of an EFS with good performance, we must set C with an appropriate value. However, under evolving streaming data, it is very difficult to decide which value of neuros is enough to represent the data distribution. In contrast, SOINN does not need a predefined number of neuros. From Fig. 2(c) and (d), we can see SOINN automatically learns the neuros by setting the value of λ, and $\text{age}_{\max}$, and, as the value of λ and $\text{age}_{\max}$ becomes smaller, the number of obtained neuros decreases. The prediction result, shown in the third line of Table I, indicates that SOINN can learn neuros with a good prediction ability when $\lambda = 200$ and $\text{age}_{\max} = 50$. However, the next line of the table shows that the EFS obtained by SOINN has a poor prediction performance when $\lambda = 100$ and $\text{age}_{\max} = 25$. The reason is shown in Fig. 2(c) and (d). From Fig. 2(d), we can see the data distribution is well represented by neuros. However, because the λ and $\text{age}_{\max}$ are set with smaller values, some neuros that represent the data distribution of various areas are deleted, resulting in the data distribution not being well represented by neuros. Similar to SOINN, TN also does not need predefined numbers of neuros. From 2(e) and 2(f), we can see the neuros decrease with the value of λ and, as $\text{age}_{\max}$ becomes smaller, the number of obtained neuros decreases. However, the last two lines of Table I show the EFS obtained by TN has a good prediction performance. The reason is shown in Fig. 2(e) and (f). From these, we can see the data distribution is well represented by neuros, when $\lambda = 200$ and $\text{age}_{\max} = 50$ or $\lambda = 100$ and $\text{age}_{\max} = 25$. So, the result proves that the robustness of these two parameters is improved using our proposed algorithm. Fig. 2(a) and (b) shows that some neuros are not located in the true data distribution because of noisy neuros. However, this phenomenon does not exist in Fig. 2(c)–(f), demonstrating that the neuros obtained by SOINN and TN are more robust to noisy neuros than FCM due to the integration of topology learning. In summary, the larger the parameters "λ" and "$\text{age}_{\max}$," the more neuros that can be sustained. A larger number of remaining neuros will result in a better prediction result, but it also leads to the overfitting problem.

B. Concept Drift

Another advantage of TLFRNN is that concept drift can be solved by our proposed TN algorithm. Hence, to illustrate the

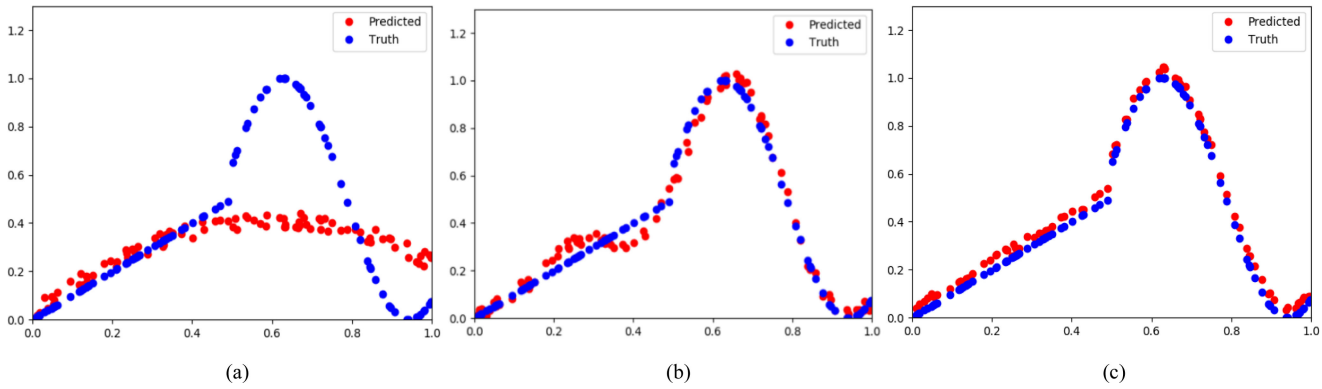

Fig. 3. Regression learning results on the artificial dataset (two models). (a) FCM. (b) SOINN. (c) FSOINN.

advantage of this, we use the following datasets:

$$\begin{cases} y(t) = x(t)\, t \text{ in } [0-400] \\ y(t) = 0.5\cos(10x(t)) + 0.5t \text{ in } (400-800] \end{cases} \tag{40}$$

where $x(\mathrm{t})U(0, 0.5)$. The artificial dataset was generated by two models. As with the last experiment, we also add 10% noise to the dataset, and the noise is distributed over the whole dataset. Fig. 6 shows the neuros in a different time-window. In this experiment, the length of the time-window is 200, i.e. $\lambda = 200$. The parameter $\mathrm{age}_{\max} = 50$.

As in the last experiment, we still use FCM, SOINN, and TN to obtain the neuros based on the dataset (40), transform these neuros into EFS by setting the smooth parameter to 0.2, and finally compare the prediction results of the EFS that have been learned through the different algorithms. As for the parameters of FCM and SOINN, we set $C = 20$ and $m = 2$ in FCM. The reason for setting $C = 20$ is that TN obtains about 40 neuros when the learning process finishes. To compare this with TN, the values of $\lambda = 200$ and $\mathrm{age}_{\max} = 50$ in SOINN match the values in TN.

Fig. 3 compares the results. From Fig. 3(a), we can see it is difficult for FCM to learn an EFS with good prediction results. The reason for this is it is difficult for FCM to learn suitable neuros to represent the data distribution from a dataset with concept drift. From 3(b), we can see SOINN obtains better results than FCM. The reason for this is SOINN has an incremental learning mechanism, so the neuros learned by SOINN will be updated to fit the new data distribution by sequentially processing the data. However, the outdated neuros cannot be deleted by SOINN, resulting in the prediction results of some neuros containing serious errors. The proposed TN is more accurate than FCM and SOINN. This is because not only is TN capable of detecting and adapting to concept drift, and it is also more robust to noisy data. In summary, our proposed TN can learn the neuros that represent the data distribution from evolving and noisy streaming data.

C. TLFRNN: Smooth Parameters

In the aforementioned experiment, we focus on validating the performance of the TN algorithm. We do this by proving the regression ability of TLFRNN. From the introduction, we know that the regression performance of our proposed TLFRNN is affected by the smooth parameter. However, it is difficult to set the smooth parameter manually without prior information of the streaming data. So, we propose a method to select the smooth parameter automatically $H(t)$. To substantiate the proposed TLFRNN, we first evaluate the method of selecting the smooth parameter $H(t)$.

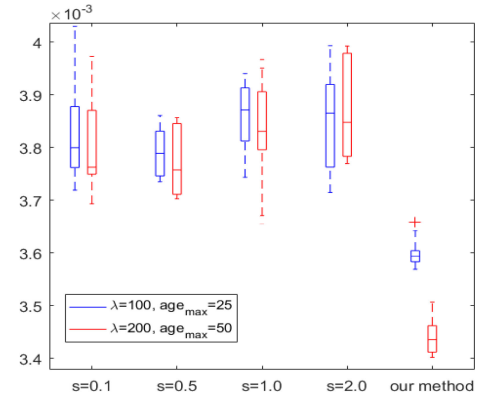

Fig. 4. RMSE based on different smooth parameter settings.

We also test the accuracy of our method with a static smooth parameter. The training sets were generated by function (39), and ARMSE was also used to measure accuracy. As for the parameters of TN, we set $\lambda = 200$, $\mathrm{age}_{\max} = 50$ and $\lambda = 100$, $\mathrm{age}_{\max} = 25$respectively. We next used the different parameter settings to obtain neuros 30 times. A comparison of the results is shown in Fig. 4.

Fig. 4 shows that our method is the most accurate and, if more neuros are generated, our method achieves better accuracy. However, this is not true with the static smooth parameter; the level of accuracy barely changes with an increase in the number of neuros. This also indicates that FSOINN is effective at reaching the optimal state.

D. TLFRNN: Artificial Dataset

This section presents the experiment results for some of the artificial datasets. The comparison experiments were carried out with nonfuzzy systems, EFSs and our proposed TELFIS. The nonfuzzy systems are fast incremental model trees with drift detection (FIMTDD) [52], adaptive model rules (AMR) [53].

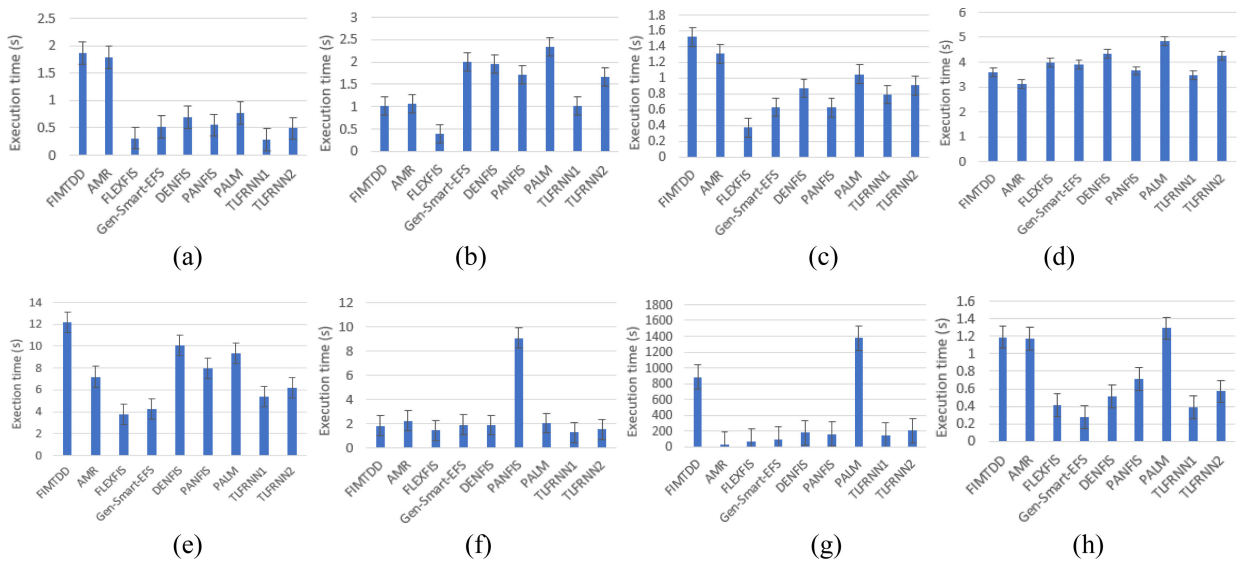

Fig. 5. Comparative results of execution time. (a) CCPP. (b) Parkinsons. (c) CPU. (d) California Housing. (e) Protein. (f) S&P Index. (g) SMEAR. (h) NN5.

TABLE II
PARAMETERS OF EACH ALGORITHM

Algorithms	Consequent Parameters Estimation Method	Number of Use-Defined Parameters
FMITDD	RLS	Learning rate: 0.01, and change probability: 0.05 The minimum number of samples in leaf: 200
AMR	RLS	Error threshold: 50 and minimum number of samples:200 the magnitude of changes that are allowed: 0.05
Gen-Smart-EFS	RLS	A scale factor δ: 1.35 A forgetting factor: 1.0
FLEXFIS	RLS	Learning rate: 0.01 Vigilance parameter: 0.3
DENFIS	RLS	Learning rate: 0.01 Distance threshold: 0.01
PANFIS	RLS	A value was chosen to control whether PANFIS augments its structure or tunes the current structure: 0.1 A value was chosen to yield more rules: 0.01
PALM	RLS	Two thresholds of rules merge: 0.5 and 0.5 An adjustment parameter which controls the membership grades :50
TLFRNN[1]	Maximum likelihood process	Frequency of neuron removal: 100 Lifetime of each edge: 25
TLFRNN[2]	Maximum likelihood process	Frequency of neuron removal: 200 Lifetime of each edge: 50

The EFSs include two types: eFRB includes FLEXFIS [16] and Gen-Smart-EFS [66] (an extension of FLEXFIS)) and eNF includes DENFIS [17], Parsimonious network based on fuzzy inference system (PANFIS) [55] and parsimonious learning machine (PALM) [56]. FIMTDD and AMR are included to demonstrate the need to use fuzzy learning. The reason for including EFSs is to demonstrate the advantages of using topology learning. Table II gives the parameters of each algorithm. The parameters of each algorithm are selected according to their own criteria within the optimal parameters. If a method applies a window-based updating strategy during the experiments, such as FIMTDD, the size of the time-window is 200. Each algorithm is also evaluated using ARMSE which is calculated using the predicted outputs and the real outputs from the online data.

The first artificial dataset is the drifting hyperplane dataset [57]. This is a well-known drift dataset for evaluating algorithms that deal with concept drift. It contains noise, gradual drifts, and nonrecurring drifts. The whole dataset consists of 10 inputs with a uniform distribution over the interval of $[0,\ 1]$ and 1 output data $y_i \in [0,\ 1]$; and there are 2000 data $(M\ =\ 2000)$ in the dataset. The dataset contains four concepts, where each concept holds 500 data. A random variate noise uniformly distributed in the interval of $[-0.1,\ 0.1]$ is injected into each output y_i (for $i\ =1, \ldots,\ M$). The value of y_i is set to 0 or 1 if its value is less than 0 or greater than 1, respectively.

A drifting Friedman's function serves as the second dataset. Friedman's function has linear and nonlinear relations between the input and output variables. The function contains five input

TABLE III
COMPARISON RESULTS ON ARTIFICIAL DATASETS (ARMSE)

	FIMTDD	AMR	Gen-Smart-EFS	FLEXFIS	DENFIS	PANFIS	PALM	TL-EFS[1]	TL-EFS[2]
Hyperplane	4.421	4.478	3.773	3.866	3.764	3.833	3.961	3.669	3.657
Friedman	2.627	3.122	2.875	2.966	2.724	2.759	2.665	2.468	2.453
Network	0.982	1.091	0.832	0.839	0.841	0.867	0.852	0.832	0.820

variables, and one output variable. The input variables are uniformly distributed over the interval of [0, 1]. To create drifting scenarios, one drifting dataset using the original Friedman's function was produced according to [58]. The dataset also has 2000 data and abrupt concept drifts. There are three points of abrupt drift in the training dataset, the first one at $\frac{1}{4}M$ data, the second at $\frac{1}{2}M$ data and the third at $\frac{3}{4}M$ data. During the first stationary period, the Friedman function is modeled. At the first and second points of abrupt drift, the modified functional dependencies are introduced and the regions R_1 and R_2 are expended. The complete description of these drifting data can be found in [59].

The third artificial dataset is the network traffic flow. We use the TCP traffic data that we collected to build an experimental dataset. We collected 12 590 traffic data points as a time-series data set. The traffic data are processed to present the amount of traffic in bytes per unit of time and represent a sample. The traffic data is aggregated with time bin 1s, that is the number of packages that arrive within the 1s time interval. To generate noise, we randomly add 1000 samples as noisy data. If it is a noisy data, the value of traffic data is set to 0.

Table III gives the results of our comparison based on artificial datasets. By comparing the results obtained from the three datasets, we can see the accuracy achieved by EFSs is higher than the nonfuzzy systems except on the Friedman dataset. Although, the results obtained by EFSs on the Friedman dataset (except for our proposed EFS) are lower than FIMTDD, the performance of EFSs is better than AMR. Therefore, the comparative results prove the advantage of fuzzy learning in processing noisy data. The comparative results also illustrate the ability of the detection mechanism of the current EFSs to adapt well to incremental drift, but not as successfully to abrupt drift. This results in FIMTDD achieving better performance than current EFSs on the Friedman dataset. In contrast, the fact that TLFRNN[1] and TLFRNN[2] obtain the best performance on the three datasets shows that the proposed EFS adapts well to abrupt or incremental drift, thereby proving the effectiveness of the topological network-based detection mechanism of concept drift. Next, comparing all fuzzy systems, we can see our proposed TLFRNN obtains better performance on all datasets whether $\lambda = 200$, $\mathrm{age}_{\max} = 50$ or $\lambda = 100$, $\mathrm{age}_{\max} = 25$. This phenomenon also proves the neuros obtained by the proposed TN algorithm are more suitable for the distribution of the actual output data. The result obtained by TLFRNN when $\lambda = 200$ and $\mathrm{age}_{\max} = 50$ is close to the result obtained by TLFRNN when $\lambda = 100$ and $\mathrm{age}_{\max} = 25$. So, this outcome proves the performance of the proposed TLFRNN is not very sensitive to λ and $\mathrm{age}_{\max}$.

TABLE IV
REAL-WORLD DATASETS

ID	*Dataset*	SAMPLE SIZE	Attributes
D1	*CCPP*	9586	5
D2	*Parkinsons*	5875	21
D3	*CPU*	8192	19
D4	*California Housing*	20640	8
D5	*Protein*	45730	9
D6	*S&P Index*	14893	1
D7	*SMEAR*	140576	43
D8	*NN5*	7350	10

E. Regression on Real-World Datasets

In this section, we use real-world datasets to further validate the performance of our algorithm. The datasets were selected from different applications with a wide range of data sizes and dimensionality. First, five datasets were taken from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php), where it was proved in [60] that combined cycle power plant (CCPP) does not contain any drift and three time-series datasets were taken from different real applications (S&P index, SMEAR, and NN5). The S&P index dataset is a dynamic real-world financial dataset on the Yahoo finance website which contains data on the period from January 3, 1950 to March 12, 2009. This dataset is highly dynamic as evidenced by the two peaks and the valley in the data trend around 2000, 2003, and 2007. The SMEAR dataset comprises a set of 30-min-interval environment observations collected from the SMEAR II station. There are many uncertain data in this dataset. The NN5 datasets comprise 111 time series with 735 observations originating from daily withdrawals at 111 cash machines in England. However, we only combine the first ten time series of NN5 into D8. Table IV gives the type of datasets and the sample size.

The final comparison experiments were carried out based on all the algorithms which are used in the earlier section. In real-world experiments, we use the first 2000 samples as a batch dataset, and then we use a grid search method [67] with cross-validation to select the parameters, where λ values in [1000], [500], [200], [100], [50], and $\mathrm{age}_{\max}$ values in [500], [200], [100], [50]. Furthermore, the size of the time-window is 200 for FMITDD. Table V gives the results of real-world experiments. The experiments demonstrate that TLFRNN outperforms the other algorithms on most datasets except on the "Parkinsons"

TABLE V
COMPARISON RESULTS ON REAL-WORLD DATASETS (ARMSE)

	D1	D2	D3	D4	D5	D6	D7	D8
FMITDD	0.035	**0.068**	0.039	0.140	0.254	0.064	0.023	0.945
AMR	0.034	0.069	0.039	0.143	0.255	0.089	0.014	0.952
PALM	0.053	0.075	0.042	0.139	0.245	0.047	0.033	1.003
Gen-Smart-EFS	0.048	0.071	**0.038**	0.133	0.258	0.061	0.018	0.932
DENFIS	0.051	0.070	0.042	0.180	0.244	0.055	0.037	0.971
PANFIS	0.049	0.073	0.040	0.179	0.253	0.052	0.021	0.959
PALM	0.055	0.074	0.041	0.155	0.257	0.042	**0.010**	0.966
TLFRNN^1	0.033	0.082	0.039	**0.136**	0.233	**0.036**	0.047	0.933
TLFRNN^2	**0.032**	0.083	0.039	0.137	**0.232**	0.037	0.048	**0.927**

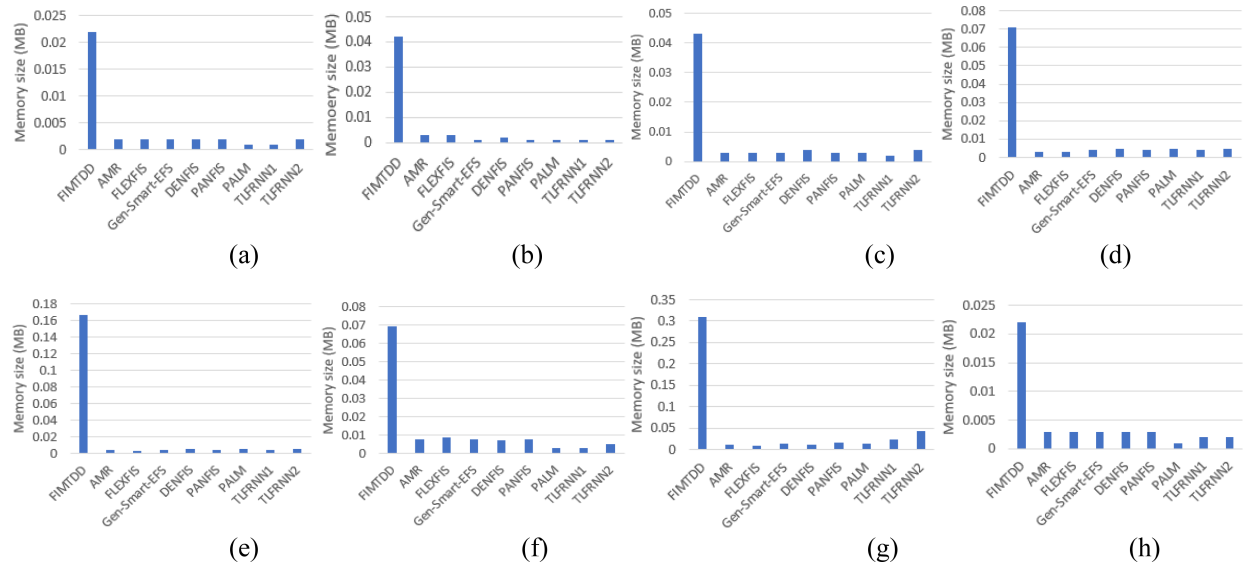

Fig. 6. Comparative results of model size. (a) CCPP. (b) Parkinsons. (c) CPU. (d) California Housing. (e) Protein. (f) S&P Index. (g) SMEAR. (h) NN5.

dataset and the SMEAR dataset. Hence, the experiment results prove the superiority of using topology learning and introducing randomness for learning EFS. The reason why TLFRNN does not achieve the best performance because the "Parkinsons" dataset is more suitably trained with a linear model. However, its nonlinear learning ability is well demonstrated on the other datasets. As TLFRNN does not achieve the best performance on the SMEAR dataset, this shows that TLFRNN still has limitations on high dimensional datasets and gives a comparable performance to nonfuzzy algorithms (FIMTDD and AMR). The results of TLFRNN^1 and TLFRNN^2 show the proposed algorithm is robust to parameters. In summary, the proposed algorithm is a more accurate alternative to current online regression algorithms for streaming data.

Next, to further evaluate the performance of our proposed TLFRNN, the mean execution time (second) was also considered in each dataset (see Fig. 5). From Fig. 5, we can see our proposed TLFRNN achieves competitive performance in almost all datasets compared with evolving nonfuzzy systems and EFSs. Hence, these experimental results illustrate the TLFRNN meets the requirement of designing a system for streaming data, i.e., time complexity cannot be too large. Finally, we compare the mean model size (MB). The comparative results are shown in Fig. 6. From Fig. 6, the amount of memory used by different systems was similar for nearly all the datasets, except for the FMITDD algorithm.

VI. CONCLUSION

In order to improve the performance of EFN systems for streaming data regression, in this article, we propose a novel EFN system, called the topology learning-based evolving-fuzzy-system. In TLFRNN, to decrease the sensitivity of prediction accuracy to the system structure, first, we learnt multiple fuzzy sets and introduce a randomness layer to assign each fuzzy set a probability. Then, to make fuzzy rules model the nonlinear relationship between inputs and outputs well, we proposed a new type of fuzzy rule, and further, a new type of inference is designed. Our designed inference not only fits the nonlinear function well, it also considered the random and fuzzy information simultaneously. Furthermore, the inference of TLFRNN was only influenced by one parameter. This parameter does not

need to be learned because it can be decided by a maximum likelihood process. To handle concept drift, a topology network-based mechanism for concept drift is introduced. As the topology network can accurately represent the real data distribution, TLFRNN can adapt to new data distributions easily and rapidly without detecting concept drift. The experiment results showed that our proposed algorithm has better performance in relation to streaming data regression.

In future work, in order to improve the prediction accuracy of our proposed TLFRNN, a mechanism for overfitting prevention will be introduced. An outlier detection technique should also be introduced to find unusual distributions.

REFERENCES

[1] X. Gu, Q. Shen, and P. P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, to be published.

[2] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man. Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.

[3] H. Yu, J. Lu, and G. Zhang, "An online robust support vector regression for data streams," *IEEE Trans. Knowl. Data Eng.*, to be pulished.

[4] H. Yu, J. Lu, and G. Zhang, "Continuous Support vector regression for nonstationary streaming data," *IEEE Trans. Cybern.*, to be published.

[5] A. Liu, J. Lu, and G. Zhang, "Concept drift detection: Dealing with missing values via fuzzy distance estimations," *IEEE Trans. Fuzzy Syst.*, to be publisehd.

[6] J. Lu, A. Liu, Y. Song, and G. Zhang, "Data-driven decision support under concept drift in streamed big data," *Complex Intell. Syst.*, vol. 6, no. 1, pp. 157–163, 2020.

[7] E. Lughofer, "Evolving fuzzy systems—methodologies," in *Advanced Concepts and Applications (Studies in Fuzziness and Soft Computing series)*, vol. 266. New York, NY, USA: Springer, 2011.

[8] P. Angelov "*Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*. Heidelberg, Germany: Physica-Verlag, 2002.

[9] P. Angelov and R. Buswell, "Identification of evolving fuzzy rule-based models," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 5, pp. 667–677, Oct. 2002.

[10] P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi–Sugeno fuzzy models," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 22–25, pp. 1068–1073.

[11] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Trans Fuzzy Syst*., no. 16, pp. 1462–1475, Dec, 2008.

[12] X. Gu, "Multi-Layer Ensemble Evolving Fuzzy Inference System," *IEEE Trans. Fuzzy Syst.*, to be published.

[13] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "PANFIS: A novel incremental learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 55–68, Jan, 2014.

[14] X. Gu, P. Angelov, and H. J. Rong, "Local optimality of self-organising neuro-fuzzy inference systems," *Inf. Sci.*, vol. 503, pp. 351–380, 2019.

[15] Y. Song, J. Lu, H. Lu, and G. Zhang, "Fuzzy clustering-based adaptive regression for drifting data streams," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 3, pp. 544–557, Mar. 2020.

[16] E. D. Lughofer, "FLEXFIS: A robust incremental learning approach for evolving Takagi-Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1393–1410, Dec. 2008.

[17] N. K. Kasabov and Qun Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[18] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014.

[19] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2018.

[20] J. Lu, J. Xuan, G. Zhang, and X. Luo, "Structural property-aware multilayer network embedding for latent factor analysis," *Pattern Recognit*., vol. 76, pp. 228–241, 2018.

[21] J. H. Aladi, C. Wagner, J. M. Garibaldi, and A. Pourabdollah, "On transitioning from type-1 to interval type-2 fuzzy logic systems," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2015, pp. 1–8.

[22] D. Pekaslan, C. Wagner, and J. M. Garibaldi, "ADONiS - Adaptive online non-singleton fuzzy logic systems," *IEEE Trans. Fuzzy Syst.* vol. 28, no. 10, pp. 2302–2312, 2020.

[23] S. Furao and O. Hasegawa, "An incremental network for online unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, 2006.

[24] C. A. Astudillo and B. J. Oommen, "Topology-oriented self-organizing maps: A survey," *Pattern Anal. Appl.*, vol. 17, no. 2, pp. 223–248, 2014.

[25] J. Lu, H. Zuo, and G. Zhang, "Fuzzy Multiple-source Transfer Learning," *IEEE Trans. Fuzzy Syst.*, to be published.

[26] P. Angelov and X. Zhou, "Evolving fuzzy systems from data streams in real-time," in *Proc. Int. Symp. Evol. Fuzzy Syst.*, 2006, pp. 29–35.

[27] P. Angelov, "Evolving Takagi-Sugeno fuzzy systems from data streams (eTS+)," in *Evolving Intelligent Systems: Methodology and Applications*, P. Angelov, D. Filev, and N. Kasabov, Eds., New York, NY, USA: Willey, Apr. 2010, pp. 21–50.

[28] R. J. Bao, H. J. Rong, P. P. Angelov, B. Chen, and P. K. Wong, "Correntropy-based evolving fuzzy neural system," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 3, pp. 1324–1338, 2018.

[29] G. Wang, T. Zhou, K.-S. Choi, and J. Lu, "A Deep-Ensemble-level-based interpretable Takagi-Sugeno-Kang fuzzy classifier for imbalanced data," *IEEE Trans. Cybern.*, to be published.

[30] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey," *Inf. Sci.*, vol. 490, pp. 344–368, 2019.

[31] 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.

[32] D. Pekaslan, J. M. Garibaldi, and C. Wagner, "Exploring subsethood to determine firing strength in non-singleton fuzzy logic systems," in *Porc. IEEE Int. Conf. Fuzzy Syst.*, 2018, pp. 1–8.

[33] F. Liu, G. Zhang, and J. Lu, "Multi-source heterogeneous unsupervised domain adaptation via fuzzy-relation neural networks," *IEEE Trans. Fuzzy Syst.*, to be published, doi: 10.1109/TFUZZ.2020.3018191.

[34] B. Kosko, *Neural Networks and Fuzzy Systems*, Englewood Cliffs, NJ, USA: Prentice-Hall, 1992.

[35] E. Czogala and J. Leski, *Fuzzy and Neuro-Fuzzy Intelligent Systems*. Heidelberg, Germany: Physica-Verlag, 2000.

[36] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy C-means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 370–379, Aug. 1995.

[37] M. S. Yang, "A Survey of Fuzzy Clustering," *Math. Comput. Model.*, vol. 18, pp. 1–16, 1993.

[38] M. T. Morf, T. Kailath, and L. Ljung, "Fast algorithms for recursive identification," in *Proc. Congr. Decis. Control*, 1976, pp. 916–921.

[39] N. K. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst. Men, Cybern. B, Cybern.*, vol. 31, no. 6, pp. 902–918, Dec. 2001.

[40] N. S. H.-J. Rong, G.-B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets Syst*., vol. 157, no. 9, pp. 1260–1275, 2006.

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

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

[43] H. Kwakernaak, "Fuzzy random variables-I. definitions and theorems," *Inf. Sci.* vol. 15, no. 1, pp. 1–29, 1978.

[44] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Netw.*, vol. 20, no. 8, pp. 893–903, 2007.

[45] H. Yu, J. Lu, G. Zhang, and D. Wu, "A dual neural network based on confidence intervals for fuzzy random regression problems," in *Proc. IEEE Int. Conf. Fuzzy Syst*., Jul. 2018, pp. 1–8.

[46] H. Yu, J. Lu, and G. Zhang, "Online topology learning by a gaussian membership-based self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 3947–3961, Oct. 2020.

[47] E. Lughofer and P. Angelov, "Handling drifts and shifts in online data streams with evolving fuzzy systems," *Appl. Soft Comput. J.*, vol. 11, no. 2, pp. 2057–2068, 2011.

[48] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M. J. Er, "An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 5, pp. 1175–1192, 2017.

[49] W. Q. Zhao, K. Li, and G. W. Irwin, "A new gradient descent approach for local learning of fuzzy neural models," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 30–44, Feb. 2013.
[50] J. Gama, *Knowledge Discovery From Data Streams*. London, U.K.: Chapman and Hall, 2010.
[51] A. Osojnik, P. Panov, and S. Džeroski, "Tree-based methods for online multi-target regression," *J. Intell. Inf. Syst.*, vol. 50, no. 2, pp. 315–339, 2018.
[52] E. Ikonomovska, J. Gama, and S. Džeroski, "Learning model trees from evolving data streams," *Data Mining Knowl. Discovery*, vol. 23, no. 1, pp. 128–168, 2011.
[53] J. Duarte, J. Gama, and A. Bifet, "Adaptive model rules from high-speed data streams," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 3. pp. 1–22, 2016.
[54] E. Lughofer, C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems," *Evol. Syst.*, vol. 6, no. 4, pp. 269–292, 2015.
[55] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "PANFIS: A novel incremental learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 55–68, Jan. 2014.
[56] M. M. Ferdaus, M. Pratama, S. G. Anavatti, and M. A. Garratt, "PALM: An incremental construction of Hyperplanes for data stream regression," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 11, pp. 2115–2129, Nov. 2019.
[57] A. Shaker and E. Hullermeier, "IBLStreams: A system for instance-based classification and regression on data streams," *Evol. Syst.*, vol. 3, no. 4, pp. 235–249, 2012.
[58] J. H. Friedman, "Multivariate adaptive regression splines with discussion," *Ann. Statist.*, vol. 19, no. 1, pp. 67–82, Mar. 1991.
[59] E. Ikonomovska, "Algorithms for learning regression trees and ensembles on evolving data streams," Ph.D. dissertation, Jozef Stefan Int. Postgraduate Sch., Ljubljana, Slovenia, Oct. 2012.
[60] L.-Y. Wang, C. Park, K. Yeon, and H. Choi, "Tracking concept drift using a constrained penalized regression combiner," *Comput. Statist. Data Anal.*, vol. 108, pp. 52–69, 2017.
[61] E. Lughofer, M. Pratama, and I. Skrjanc, "Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 1854–1865, Aug. 2018.
[62] 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.
[63] C. F. Juang, T. C. Chen, and W. Y. Cheng, "Speedup of implementing fuzzy neural networks with high-dimensional inputs through parallel processing on graphic processing units," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 717–728, Aug. 2011.
[64] G. Der Wu and P. H. Huang, "A maximizing-discriminability-based self-organizing fuzzy network for classification problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 362–373, Apr. 2010.
[65] W. Y. Cheng and C. F. Juang, "A fuzzy model with online incremental SVM and margin-selective gradient descent learning for classification problems," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 2, pp. 324–337, Apr. 2014.
[66] C. F. Juang and K. J. Juang, "Circuit implementation of data-driven TSK-type interval type-2 neural fuzzy system with online parameter tuning ability," *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4266–4275, May 2017.
[67] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2012.

Hang Yu is working toward the Ph.D. degree in software engineering at the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia.

He is a member of the Decision Systems and e-Service Intelligence (DeSI) Research Laboratory, Centre for Artificial Intelligence, University of Technology Sydney. His research interests include streaming data mining, concept drift, and fuzzy systems.

Jie Lu (Fellow, IEEE) received the Ph.D. degree in information system from Curtin University, Bentley, WA, Australia, in 2000.

She is currently a Distinguished Professor and the Director of Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, NSW, Australia. She has published six research books and over 450 papers in refereed journals and conference proceedings and has won 11 Australian Research Council (ARC) grants and completed many industry projects. Her main research interests are in the areas of fuzzy transfer learning, concept drift, decision support systems, and recommender systems.

Prof. Lu was the recipient of various awards, such as the IEEE Transactions on Fuzzy Systems Outstanding Paper Award (2019), and the Australian Most Innovative Engineer Award (2019). She is an IFSA Fellow and Australian Laureate fellow. She is the Editor-In-Chief for Knowledge-Based Systems (Elsevier) and Editor-In-Chief for the International Journal of Computational Intelligence Systems. She has delivered over 25 keynote speeches at international conferences and chaired 15 international conferences.

Guangquan Zhang received the Ph.D. degree in applied mathematics from Curtin University, Bentley, WA, Australia, in 2001.

He is currently an Associate Professor and the Director of the Decision Systems and e-Service Intelligent Research Laboratory, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. He has authored five monographs, five textbooks, and 450 papers in Artificial Intelligence Journal, Machine Learning Journal, IEEE TRANSACTIONS ON FUZZY SYSTEMS and other refereed journals and conference proceedings. His research interests include fuzzy machine learning, fuzzy optimization, and machine learning and data analytics.

Dr. Zhang has won nine Australian Research Council Discovery Project grants and many other research grants. He was awarded an ARC QEII Fellowship in 2005. He was a member of the editorial boards of several international journals, as a guest editor of eight special issues for IEEE Transactions and other international journals and co-chaired several international conferences and workshops in the area of fuzzy decision-making and knowledge engineering.