Information Sciences 507 (2020) 172–184

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Learning data streams online — An evolving fuzzy system approach with self-learning/adaptive thresholds

Dongjiao Ge, Xiao-Jun Zeng[*]

School of Computer Science, The University of Manchester, Oxford Road, Manchester, M13 9PL, UK

A R T I C L E   I N F O

Article history:
Received 8 February 2019
Revised 6 August 2019
Accepted 11 August 2019
Available online 12 August 2019

Keywords:
Evolving fuzzy systems
Data streams
Online learning

A B S T R A C T

Recognizing the weakness of the existing evolving fuzzy systems (EFSs) where the selection and determination of thresholds for the structure and parameter learning are completely relied on the trial and error strategy, this paper proposes a novel and symmetrical approach with self-learning/adaptive thresholds (EFS-SLAT) for EFSs to overcome such a fundamental weakness. Departing from the common but implicit assumption in the existing EFS approaches that the thresholds are fixed parameters throughout the learning process, EFS-SLAT treats the thresholds as dynamical parameters which are varying with the evolution of systems being learned. By utilizing the online training errors as an indicator to reflect the underfitting and overfitting state of an EFS model, the proposed EFS-SLAT selects and adjusts the values of threshold parameters automatically and dynamically based on the evolving speed and nonlinear degree of the EFS. By testing EFS-SLAT on several well-known benchmark examples, and comparing it with many state-of-the-art approaches, EFS-SLAT is verified to be capable of giving preferable results.

1. Introduction

1.1. State-of-the-art & motivations

In the Information Explosion era, many data collected from our everyday life take the form of data streams, e.g., data generated by social media, sensor network and financial transactions, etc. Data streams usually come in high speed and often accompany with concept drift (nonstationary) phenomenon. Hence, the data stream mining methodologies should manage to learn from the rapidly arriving data, to discover the underlying knowledge and to handle the concept drift behind the data online.

As a family of effective and promising data stream regression approaches, evolving fuzzy systems (EFSs) have received a great deal of attention. EFSs are fuzzy rule based systems with self-learning structures and parameters. EFSs have two remarkable characteristics: 1) EFSs are equipped with a real-time evolving structure that can capture the data dynamics and concept drifts; 2) EFSs can present the knowledge learned online in an accurate, transparent and interpretable way [22,30]. Generally speaking, the existing approaches in EFSs usually consist of two major learning modules: *structure evolving* including rule generation and simplification (merging and pruning) and *parameters updating* containing antecedent and consequent parameters updating. The general framework of EFSs could be depicted in Fig. 1. Existing works have shown

[*] Corresponding author.
E-mail addresses: dongjiao.ge@manchester.ac.uk (D. Ge), x.zeng@manchester.ac.uk (X.-J. Zeng).

https://doi.org/10.1016/j.ins.2019.08.036
0020-0255/© 2019 Elsevier Inc. All rights reserved.

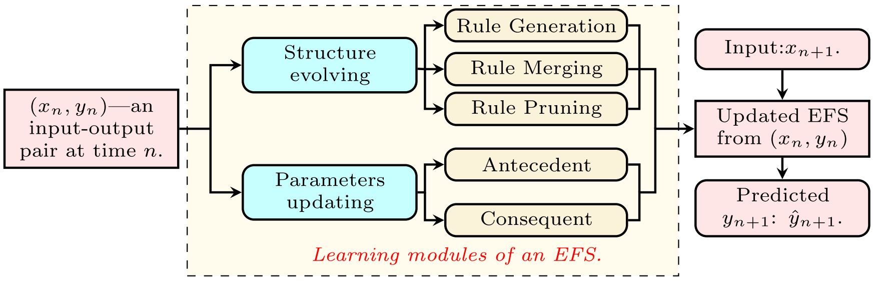
(x_n, y_n)—an input-output pair at time n.
Structure evolving
Parameters updating
Rule Generation
Rule Merging
Rule Pruning
Antecedent
Consequent
Learning modules of an EFS.
Input: x_{n+1}.
Updated EFS from (x_n, y_n)
Predicted y_{n+1}: $\hat{y}_{n+1}$.

Fig. 1. General framework of EFSs learning.

significantly more interest in investigating and setting criteria to realize structure learning than in inducting formulas for parameters updating. For this reason, the remaining part of this section focuses on summarizing and analyzing the research about the structure learning of EFSs.

Rule generation, which controls the system expanding, plays an important role in structure evolving module. Different criteria have been set to control the process of rule growth. For those criteria, the essence lies in measuring whether the new data is situated within the existing clusters. The rule generation criteria have a general form:"*If a certain metric is larger or smaller than a threshold, then a new rule should be generated.*" One typical example is the *firing strength* [9,19,29,31,35]. It is a straightforward application of the values of the firing strengths, and helps to generate a new rule once the firing strengths of all the existing rules are lower than a threshold. There are also other criteria, e.g., *distance based criterion* [6,9–11,17,20,21,26,32], *potential* [3], *data quality* [39,41] and *arousal index* [18] based criteria, etc.

Rule merging and *pruning* are two sub-modules that consist of the rule simplification module. Similar to *rule generation*, both of these two rule simplification modules control the system shrinking based on some criteria. The rule merging approaches can both avoid the rule conflict and simplify the rule base at the same time, by means of combining similar rules. The form of a rule merging criterion should be: "*If the similarity between two rules are larger than a threshold, then these two rules should be merged.*" These similarity measures of the fuzzy rules could be largely divided into set theory and geometric theory based methods [43]. Set theory based measures compare the proportion of the intersection of two clusters with the union of these two clusters. Examples could be found in [47] and [8]. The geometric measures are distance based measures which aim to compare the similarity of fuzzy sets with the shapes and/or positions of the membership functions, see [30,34,38,41] for example.

Unlike the *rule merging* approaches, *rule pruning* module set criteria and remove those negligible rules to assist in keeping a clean rule base. The criteria used for rule pruning are by the aim of judging whether the rules are negligible. These criteria also have an uniform expression: "*If a certain metric is smaller/larger than a threshold, then a rule should be regarded as redundant and be removed.*" The frequently used metric are, for example: *utility* [1,33,36] and *age* [25,44] which judge those seldom activated rules (or rules which have not been used for a long time) as unimportant rules. They decide to delete a redundant rule when its *utility*/*age* is smaller/larger than the threshold. Other important criteria are *population/density* [2,9], *rule influence* [38,42] and *rule importance* [19], etc. More specific introduction of EFSs structure learning methods can be found from some excellent summaries e.g. [22–24].

Composed by these structure evolving modules (*rule generation, merging or pruning*) discussed above, the existing state-of-the-art methodologies have greatly improved the development of EFSs. However, some shortcomings and limitations still need to be properly solved.

- Obviously, each of these structure evolving modules is required to set a predefined threshold, no matter which criteria are used. The most common way to select them is the trail and error approach with a vast number of numerical experiments. However, these predefined thresholds are difficult to guess when there is no prior knowledge of the data. Besides, the fixed thresholds can hardly fit the drift and shift of different data series. Although Ge and Zeng [15] has given a self-learning approach for rule generation threshold, this issue is not really solved as the rule simplification threshold is still selected offline.
- The problem of "Whether a certain structure evolving is really necessary?" is only answered using the information contained in the input, due to the structure evolving criteria are usually determined by the inputs. The information of the outputs is seldom considered. Therefore, it is hard to infer whether the EFSs could provide better performance by carrying out these structure evolving procedures without, or with, a little information from the output; thus, this may cause some false alarms of structure evolving.

1.2. Our approach

Based on our preliminary research [15], which allows a time varying threshold to control the rule generation, we investigate and propose an EFS with self-learning/adaptive thresholds (EFS-SLAT). This will control the system structure evolving, including both rule generation and simplification, relying on the risks of overfitting and underfitting demonstrated by the

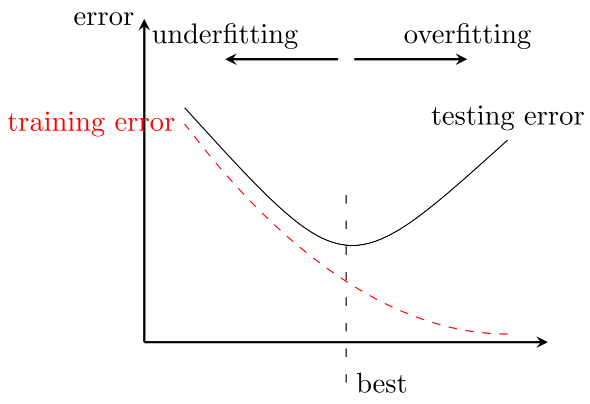

Fig. 2. Underfitting vs overfitting.

training errors. To be more specific, the EFS-SLAT adjusts the speed of evolving based on the relationship between overfitting, underfitting, training error and testing error presented by Fig. 2 (see also [46]). In Fig. 2, the underfitting corresponds to high bias and low variance, while the overfitting corresponds to the opposite side; the "best" is known as the "bias-variance tradeoff", which is the intersection of the bias and variance. A similar approach utilizing the Fig. 2 to tune the model structure can be found from [4]. Utilizing Fig. 2, EFS-SLAT could make up the shortcomings mentioned in the Section 1.1 from the following aspects:

- The dynamic thresholds, which control rule generation and reduction speed, are defined to evolve as functions of the cumulative absolute training error with a gradual forgetting. Rule generation is accelerated/ decelerated based on the risk of underfitting/overfitting reflected by the big/tiny training error. Rule simplification speed is tuned to be fast or slow with the same trend of the rule generation speed, in order to balance the possible side-effects (overfitting or underfitting) caused by the too-fast or too-slow speed of rule generation.
- To make sure that the EFSs are evolved/updated in order to meet the requirement that the updated system should give a better track of the latest information, EFS-SLAT decides to execute the structure change or existing rule updating by voting the evolving approaches which can achieve better training error for the latest training data. This approach enables the information contained in the output to be utilized in order to guide the system towards evolving in the proper direction.

The remainder of the paper is organized as follows: Section 2 states the online learning tasks of EFS-SLAT. Thereafter, the proposed learning approaches including rule generation, reduction, and parameters updating are presented in Section 3. Numerical examples and conclusions are provided in Sections 4 and 5, respectively.

2. Problem statement

The problem which needs to be solved by the proposed method EFS-SLAT is to identify the Takagi-Sugeno (T-S) fuzzy systems online. A T-S fuzzy system F is constituted by a group of the following T-S fuzzy rules r_i (1).

$$r_i : \text{IF } x_1 \text{ is } \Gamma_{i,1}, \ldots, \ x_m \text{ is } \Gamma_{i,m}, \ \text{THEN } y_i = \psi_0 + \psi_1 x_1 + \ldots + \psi_m x_m, \tag{1}$$

where $i = 1, 2, \ldots, K$, K is the rule number, $x = (x_1, x_2, \ldots, x_m) \in [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_m, b_m] \subset \mathbf{R}^m$ is the input, $(\psi_0, \psi_1, \ldots, \psi_m)$ is the consequent parameter and $y_i \in \mathbf{R}$ is the output of the rule r_i. The membership function of the fuzzy set $\Gamma_{i,j}$ is $\mu_{i,j}(x_j)$ with expression:

$$\mu_{i.j}(x_j) = \exp\left\{-\frac{(x_j - c_{i,j})^2}{2\sigma_{i,j}^2}\right\}, \tag{2}$$

in which $c_{i,j}$ and $\sigma_{i,j}$ are the cluster center and radius of $\Gamma_{i,j}$, respectively. The overall output of the fuzzy system F could be computed by:

$$y = \sum_{i=1}^{K} \theta_i(x) y_i / \left(\sum_{i=1}^{K} \theta_i(x) y_i\right), \tag{3}$$

where $\theta_i(x) = \prod_{j=1}^{m} \mu_{i,j}(x_j)$ is the firing strength.

In order to identify a T-S fuzzy system, the task for EFS-SLAT is to learn the rule numbers K, antecedent parameters which include the cluster centers $c_{i,j}$ and the radiuses $\sigma_{i,j}$, and consequent parameters $\psi_{i,j}$ from the data streams.

3. Learning approaches of EFS-SLAT

Fuzzy system identification approaches of EFS-SLAT are specifically explained in this section. Based on the dynamic and time-varying property of EFS-SLAT, the fuzzy system learned by EFS-SLAT has its structure (rule numbers) and all the threshold parameters changing over time. This phenomenon naturally leads to the problem of how to identify or update the rule base and the parameters. Depended on the discussion in Section 1.1 and 1.2, we present the five learning modules, *rule generation, antecedent parameters updating, rule merging, rule pruning* and *consequent parameters updating*, in the Section 3.1–3.3.

3.1. Rule generation and antecedent parameters updating

Rule generation depends on the novelty content of the new sample, which means that new rule should be generated when the old rule base is insufficient to cover the information of the new data. On the contrary side, the old rule base should be updated using the new sample, when new data are located within the range of the existing clusters.

Assume the new input-output pair to be (x_n, y_n). The firing strength can indicate the membership degree of a data point covered by a certain cluster, according to which the firing strength is used as the major metric to measure the novelty content with regard to the new input. Besides, to avoid unnecessary rules being generated when the (x_n, y_n) is of very low firing strength, we compare $|e_n^{old}|$ and $|e_n^a|$. Because EFS-SLAT is a one-pass approach which was established based on the assumption of no historical data being remembered, so only one single pair (x_n, y_n) is used to compute these two factors. The $|e_n^{old}| = |y_n - F^{old}(x_n)|$ is the online training error computed by the old system with no new rules been generated; while $|e_n^a| = |y_n - F^{new}(x_n)|$ is calculated using the new system that has been generated a new rule. As $|e_n^{old}| \geq |e_n^a|$ indicates the existing system can depict the new data pair well, so new rules should not be generated to keep the system in a low complexity while minimizing the training error. To summarize, the final decision of whether a new rule should be generated is determined from two aspects: 1) The new sample has very low *firing strength*; and 2) The training error could be substantially declined without unnecessary increasing of the complexity. The exact method is shown in Criterion 3.1.

Criterion 3.1 Rule generation method:. If $\forall i = 1, 2, \ldots, K$ such that $\theta_i(x_n) < \epsilon_n^a$ and $|e_n^{old}| \geq |e_n^a|$ holds, then generate a new fuzzy rule r_{K+1}. The cluster center is $c_{K+1} = x_n$, cluster radius is $\sigma_{K+1} = \|x_n - c_{i'}\|/\sqrt{-2\log(\sup \epsilon_n^a)}$, consequent parameters are $\psi_{K+1} = \psi_{i'}$, where $i' = \arg\min_{i=1,2,\ldots,K} \|x_n - c_i\|^2$.

Note that, even if a new rule were to be generated due to false alarms (noise or outliers), the generated cluster will be kept as a tiny size and will be finally removed by the rule pruning approach. On the opposite side of rule generation, assume there exists a rule r_{i^*} such that the firing strength regarding to x_n is large. In this case, x_n should be regarded as a data sample in the corresponding cluster of rule r_{i^*}, hence, the antecedent parameters of r_{i^*} should be adjusted based on the Criterion 3.2.

Criterion 3.2 Rule updating method:. If $\exists i^* \in \{1, 2, \ldots, K\}$ such that $i^* = \arg\max_{i=1,\ldots,K} \theta_i(x_n)$ and $\theta_{i^*}(x_n) \geq \epsilon_n^a$, then adjust the center and the radius of rule r_{i^*} by (4) and (5):

$$c_{i^*}^{new} = c_{i^*}^{old} + \frac{1}{N_{i^*}^{old} + 1}(x_n - c_{i^*}^{old}), \tag{4}$$

$$(\sigma_{i^*,j}^{new})^2 = (\sigma_{i^*,j}^{old})^2 + \frac{(x_{n,j} - c_{i^*,j}^{new})^2 - (\sigma_{i^*,j}^{old})^2}{N_{i^*}^{old} + 1} + \frac{N_{i^*}^{old}(c_{i^*,j}^{new} - c_{i^*,j}^{old})^2}{N_{i^*}^{old} + 1}. \tag{5}$$

Similar method could also be found in [1,2,14,15,17,32,41]. The number of samples within cluster formed by r_{i^*} should be updated by $N_{i^*}^{new} = N_{i^*}^{old} + 1$.

Both Criterions 3.1 and 3.2 are controlled by the threshold ϵ_n^a. Differ from setting ϵ_n^a as a fixed predefined parameter, EFS-SLAT learns and update ϵ_n^a dynamically based on the cumulative online training absolute error $\tilde{e}_n = \sum_{i=1}^{n} \lambda^{n-i}|e_i|$ ($e_i = y_i - \hat{y}_i^{train}$, $\hat{y}_i^{train} = F^{updated}(x_i)$, $F^{updated}$ is the updated system obtained when all the evolving modules are processed using (x_i, y_i), $\lambda \in [0, 1]$ is the forgetting factor). The adjusting mechanism of ϵ_n^a is:

- Tune up ϵ_n^a when $\tilde{e}_n$ grows. Large cumulative online training error $\tilde{e}_n$ indicates the model is underfitting, hence an insufficient number of fuzzy rules are used. New rules should be generated quickly to catch up with the rapid changes of the data stream, and to help to improve the performance. This task could be achieved by increasing ϵ_n^a.
- The ϵ_n^a should be set small, when $\tilde{e}_n$ has a small value. Although $\tilde{e}_n$ could not exactly reflect whether the overfitting really happens, it can act as an indicator to present the risk for overfitting (i.e. the smaller the $\tilde{e}_n$, the higher risk for the overfitting). Therefore, it is appropriate to slow down the speed for rule generation when $\tilde{e}_n$ declines.

Therefore, the ϵ_n^a is designed to be the monotonically increasing function of $\tilde{e}_n$ in order to achieve the above mechanism. The logistic function $g(x) = 1/(1 + \exp\{-x\})$, which is a frequently used monotonically increasing function for clamping the data into a certain range in machine learning, is used to construct the threshold variation function. Considering the

k-sigma rule for the multivariate Gaussian distribution, and the 1-sigma which forms the core of a Gaussian fuzzy set with its explanatory term, the upper bound of ϵ_n^a is set to be $\exp\{-0.5\}$. Besides, the natural lower bound is 0, which means no rules need to be generated. To summarize, $\epsilon_n^a \in [0, \exp\{-0.5\})$ is calculated through:

$$\epsilon_n^a = 2e^{-0.5}\left\{g(-\tilde{e}_n) - \frac{1}{2}\right\} = e^{-0.5}\left\{\frac{2}{1+\exp(-\tilde{e}_n)} - 1\right\}, \tag{6}$$

where the $\tilde{e}_n$ can be computed recursively by $\tilde{e}_n = \lambda\tilde{e}_{n-1} + |e_n|$. As a result, this recursive formula could be used to update the ϵ_n^a by (7):

$$\epsilon_n^a = e^{-0.5}\left\{\frac{2}{1+\exp(-\lambda\tilde{e}_{n-1} - |e_n|)} - 1\right\}. \tag{7}$$

It could be observed that big $|e_i|$ can make ϵ_n^a very close to its upper bound. Besides, the lower bound 0 would be touched at the beginning of learning; or it would also be reached when the data have a very simple behaviour. The $\tilde{e}_n$ would be reset as 0, once there is a new rule generated. This allows the rule generation speed to vary based on the different training phases. To ensure ϵ_n^a varying in a reasonable degree within $[0, \exp\{-0.5\})$, the centralization of the dataset, which have big values e.g. the value of each data point is larger than 10, 100 or 1000, is recommended.

3.2. Rule simplification

With the arrival of increasing number of data samples, sufficient knowledge of the data stream would be gained by the EFS-SLAT. Hence, the whole picture of the data behaviour would be reflected gradually. Under the background of online rule generation and adjusting, there should be two common phenomena: (1) Some of the constructed fuzzy rules as well as the clusters would become similar; (2) There also exist some fuzzy rules which grow to be redundant rules with tiny clusters, because they have seldom been activated since they were constructed. Both of the above cases may lead to overfitting, and rules with similar antecedent parts would cause rule conflict. To cope with these problems, a rule merging (Criterion 3.4) and rule pruning approach (Criterion 3.5) are used.

3.2.1. Merge similar rules

The similarity between two fuzzy rules $S(\cdot, \cdot)$ is calculated by the L^2 distance between the firing strengths directly. The same as [15], the formula to compute the similarity between two rules r_1 and r_2 is presented in (8):

$$S(r_1, r_2) = \frac{1}{1 + \|\theta_1(x) - \theta_2(x)\|_{L^2}}, \tag{8}$$

where $\|\theta_1(x) - \theta_2(x)\|_{L^2}$ is a multivariate integral presented in (9):

$$\|\theta_1(x) - \theta_2(x)\|_{L^2}^2 = \int_\Omega |\theta_1(x) - \theta_2(x)|^2 dx = \int_{a_1}^{b_1} \cdots \int_{a_m}^{b_m} \left(\prod_{j=1}^m \mu_{1,j}(x_j) - \prod_{j=1}^m \mu_{2,j}(x_j)\right)^2 dx_1 dx_2 \cdots dx_m. \tag{9}$$

According to the induction in [15], $\|\theta_1(x) - \theta_2(x)\|_{L^2}^2$ could be computed by (10):

$$\begin{aligned}\|\theta_1(x) - \theta_2(x)\|_{L^2}^2 = \left(\frac{\sqrt{\pi}}{2}\right)^n \Bigg\{&\prod_{j=1}^n \sigma_{1,j} G\big(\tilde{a}_{1,j}, \tilde{b}_{1,j}\big) + \prod_{j=1}^n \sigma_{2,j} G\big(\tilde{a}_{2,j}, \tilde{b}_{2,j}\big) \\ &- 2\prod_{j=1}^n \frac{\sqrt{2}\sigma_{1,j}\sigma_{2,j}}{\sqrt{\sigma_{1,j}^2 + \sigma_{2,j}^2}} \exp\left\{-\frac{(c_{1,j} - c_{2,j})^2}{2(\sigma_{1,j}^2 + \sigma_{2,j}^2)}\right\} G(\tilde{a}_j, \tilde{b}_j)\Bigg\},\end{aligned} \tag{10}$$

in which $\tilde{a}_{i,j} = (a_j - c_{i,j})/\sigma_{i,j}$, $\tilde{b}_{i,j} = (b_j - c_{i,j})/\sigma_{i,j}$, $i = 1, 2$, $\tilde{a}_j = (a_j - \frac{\sigma_{1,j}^2 c_{2,j} + \sigma_{2,j}^2 c_{1,j}}{\sigma_{1,j}^2 + \sigma_{2,j}^2}) / \frac{\sqrt{2}\sigma_{1,j}\sigma_{2,j}}{\sqrt{\sigma_{1,j}^2 + \sigma_{2,j}^2}}$, $\tilde{b}_j = (b_j - \frac{\sigma_{1,j}^2 c_{2,j} + \sigma_{2,j}^2 c_{1,j}}{\sigma_{1,j}^2 + \sigma_{2,j}^2}) / \frac{\sqrt{2}\sigma_{1,j}\sigma_{2,j}}{\sqrt{\sigma_{1,j}^2 + \sigma_{2,j}^2}}$, and $G(\alpha_1, \alpha_2)$ is the function shown in (11):

$$G(\alpha_1, \alpha_2) = \begin{cases} erf(\alpha_2) - erf(\alpha_1) & 0 \le \alpha_1 \le \alpha_2 \\ erf(-\alpha_1) + erf(\alpha_2) & \alpha_1 \le 0 \le \alpha_2 \\ erf(-\alpha_1) - erf(-\alpha_2) & \alpha_1 \le \alpha_2 \le 0, \end{cases} \tag{11}$$

where $erf(\cdot)$ is the error function (12):

$$erf(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} \exp\{-t^2\} dt = \frac{2}{\sqrt{\pi}} \int_0^x \exp\{-t^2\} dt, \tag{12}$$

Simply merging rules with small similarity $S(\cdot, \cdot)$ in formula (8) may cause inappropriate merging. This is because, when two fuzzy rules have very small radiuses and their centers are actually not “close” to each other, then the similarity between these two rules would also be of a small value. In order to avoid this, we only consider to merge those rules which are "touched". It is well known that if $X \sim N(c,\ \Sigma)$,[1] then the k-sigma rule could be used through computing the Mahalanobis distance $D(x, c) = \sqrt{(x-c)^T \Sigma^{-1} (x-c)}$. For the conventional T-S systems, the Mahalanobis distance would reduce to the standardized Euclidean distance. $D(x,\ c) \le 3$ can make sure that 99.7% of the data in the cluster are inside the enclosed ellipsoid.

Based on the above theory, we judge whether two rules are touched by testing whether one of these two clusters could have its center located within the ellipsoid $\sqrt{(X-c)^T \Sigma^{-1} (X-c)} \le 3$ of another cluster. Therefore, whether rule r_{i^*} and $r_{i'}$ are "touched" is judged by Criterion 3.3.

Criterion 3.3 Overlapping rules:. If rule r_{i^*} and $r_{i'}$ satisfy that $\theta_{i^*}(c_{i'}) \ge \exp(-9/2)$ or $\theta_{i'}(c_{i^*}) \ge \exp(-9/2)$, then r_{i^*} and $r_{i'}$ are regarded to be the overlapping rules.

Following on formula (8) and Criterion 3.3, similar fuzzy rules are merged according to the strategy shown in Criterion 3.4.

Criterion 3.4 Rule merging method:. If rule r_{i^*} is updated by Criterion 3.2, and $\exists i'' \in \{1, 2, \ldots, K\}$, $i'' \ne i^*$, such that $i'' = \arg \max_{i'' \ne i^*, i''=1,\ldots,K} \{S(r_{i^*}, r_{i''})\}$, $S(r_{i^*}, r_{i''}) > \epsilon_n^m$, r_{i^*} and $r_{i''}$ satisfy Criterion 3.3, and $|e_n^{old}| \ge |e_n^m|$ hold, then merge rule r_{i^*} with $r_{i''}$. $S(r_{i^*}, r_{i''})$ is the similarity value between r_{i^*} and $r_{i''}$, and $S(\cdot,\ \cdot) \in [0,\ 1]$.

In Criterion 3.4, $|e_n^{old}| = |y_n - F^{old}(x_n)|$ is the absolute training error computed by the system before merge, and $|e_n^m| = |y_n - F^{new}(x_n)|$ is computed by the new system when r_{i^*} and $r_{i''}$ are merged. Based on Criterion 3.4, assume the merged rule is r_{i^m}. The center and radius of r_{i^m} are c_{i^m} (13) and σ_{i^m} (14):

$$c_{i^m} = \frac{N_{i^*} c_{i^*} + N_{i''} c_{i''}}{N_{i^*} + N_{i''}}, \tag{13}$$

$$(\sigma_{i^m,j})^2 = \frac{1}{N_{i^*} + N_{i''}} \left\{ N_{i^*} (\sigma_{i^*,j})^2 + N_{i''} (\sigma_{i'',j})^2 + N_{i^*} (c_{i^*,j} - c_{i^m,j})^2 + N_{i'',j} (c_{i'',j} - c_{i^m,j})^2 \right\}, \tag{14}$$

similar method of which could also be found from [15]. Consequent parameters are $\psi_{i^m} = (N_{i^*} \psi_{i^*} + N_{i''} \psi_{i''})/(N_{i^*} + N_{i''})$, the number of samples in the merged cluster is $N_{i^m} = N_{i^*} + N_{i''}$. The updated system after merging r_{i^*} and $r_{i''}$ has both of these two rules deleted, but uses one rule r_{i^m} to replace them. Remark that after merging the rule numbers is updated by $K-1$. In a similar way to generating new fuzzy rules, Criterion 3.4 demonstrates that rule merging is also controlled by a time varying threshold parameter ϵ_n^m. EFS-SLAT learns ϵ_n^m based on the following theory:

- Large cumulative online training absolute error $\tilde{e}_n$ demonstrates the system is suffering underfitting, under which circumstance, ϵ_n^a becomes large in order to allow the structure of the system to change rapidly to capture the dynamics of the data. A multiplicity of new rules are likely to generate mini-clusters which may lead to overfitting. To shrink the rule base and prevent overfitting, ϵ_n^m should be tuned down to allow more fuzzy rules to be merged.
- Tiny cumulative online training absolute error $\tilde{e}_n$, which is accompanied with a slow rule generation procedure, shows the system can perfectly reflect the behaviour of the data. The slow rule generation speed, which is controlled by a small ϵ_n^a, needs to be balanced by a slow rule merging speed with large ϵ_n^m in case of underfitting, due to which merging rules quickly would lead to some important historical information being discarded.

Therefore, ϵ_n^m and ϵ_n^a have a close relationship with each other. Furthermore, ϵ_n^m is constructed to balance the possible side-effects caused by ϵ_n^a. As the threshold ϵ_n^m for the geometric similarity should be varying within [0,1], and 0.5 is the mean of the lower and upper bound of ϵ_n^m, so ϵ_n^m is built as a function with range (0.5,1]. Similar to the ϵ_n^a, the logistic function $g(\tilde{e}_n)$ is also used to formulate ϵ_n^m. The exact formula is shown in (15):

$$\epsilon_n^m = 1 - \left\{ g(\tilde{e}_n) - \frac{1}{2} \right\} = \frac{3}{2} - \frac{1}{1 + \exp(-\tilde{e}_n)}. \tag{15}$$

In addition, applying the recursive formula of $\tilde{e}_n$, the ϵ_n^m can be computed by:

$$\epsilon_n^m = \frac{3}{2} - \frac{1}{1 + \exp(-\lambda \tilde{e}_{n-1} - |e_n|)}. \tag{16}$$

3.2.2. Prune redundant rules

Due to the online property of EFS-SLAT, no historical data is known before the algorithm begins to learn. It is a fact that there would be some fuzzy rules that have seldom/never been activated since they were built, which makes these rules redundant and furthermore has a negative influence on the prediction accuracy. "Utility" [1,33] is the metric, which measures whether the rules are useful through considering the cumulative firing strength and the "age"[2]. This "utility" metric is used in this paper to pick those rules which need to be removed. EFS-SLAT does not throw out these redundant

[1] $N(c,\ \Sigma)$ is the m-dimensional multivariate Gaussian distribution with mean c and covariance matrix Σ.

rules directly. Alternatively, it removes these rules by merging them with their nearest clusters. The reason behind this is that these small clusters are also learned from the data, and it would have a negative influence on the learning accuracy when these clusters to be discarded directly. By adopting the merging operation, on one hand, the performance and the historical information would be effectively reserved; On the other hand, the rule number would be shrunk to relieve the computation burden. To be more specific, EFS-SLAT applies the following rule pruning criterion, Criterion 3.5.

Criterion 3.5 Rule pruning method:. Assume rule r_{i^s} with $i^s = \arg\min_{i=1,\dots,K} U_i$ is the fuzzy rule which has the smallest "utility". The "utility" of r_i could be computed by (17):

$$U_i = \sum_{t=T_i}^{T} \theta_i(x_t)/(T - T_i), \tag{17}$$

where $t = T_i, \dots, T$ is the time stamp, T_i is the time stamp when the rule r_i is generated, and the $\sum_{t=T_i}^{T} \theta_i(x_t)$ is the cumulative firing strength. If the "utility" U_{i^s} is smaller than ϵ_n^a in the Criterion 3.1, then merge it with rule $r_{\tilde{i}^s}$, where $\tilde{i}^s = \arg\min_{i \neq i^s, i=1,\dots,K} \|c_i - c_{i^s}\|$, according to Criterion 3.4.

3.3. Learning consequent parameters

Consequent parameters are learned by the weighted recursive least square method (WRLS). The objective functions are the local error functions (18),

$$E_i = \sum_{l=1}^{n} \theta_i(x_l)(y_l - \hat{y}_l^{train})^2. \tag{18}$$

These error functions (18) are optimized by the WRLS in (19) and (20). Similar methods could also be found in [9–11,20,21,45].

$$\psi_i(n) = \psi_i(n-1) + \frac{\theta_i(x_n)P_i(n-1)xe_n(y_n - \hat{y}_n^{train})}{1 + \theta_i(x_n)xe_n^T P_i(n-1)xe_n}, \tag{19}$$

$$P_i(n) = P_i(n-1) - \frac{\theta_i(x_n)P_i(n-1)xe_n xe_n^T P_i(n-1)}{1 + \theta_i(x_n)xe_n^T P_i(n-1)xe_n}, \tag{20}$$

where $xe_n = (1, x_n),\ i = 1, 2, \dots, K$.

4. Numerical examples

The evaluation of EFS-SLAT is carried out from the frequently used benchmark examples including S&P 500 closing price prediction, Mackey–Glass Chaotic time series prediction. Besides, EFS-SLAT is also tested on Delta Ailerons data from KEEL-dataset.[2] helicopter unmanned aerial vehicle streaming data [12], and MPG and Boston Housing data from UCI repository.[3] In all these examples, we allow the forgetting factor λ to vary in [0.8, 0.85, 0.9, 0.95, 0.99]. The suggested range of the forgetting factor is [0.8, 1). We use RMSE (21), NDEI (22) and MAE (23) to judge the accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i^{test})^2}, \tag{21}$$

$$NDEI = RMSE/std(y), \tag{22}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i^{test}|. \tag{23}$$

As a one-pass online approach, EFS-SLAT is also evaluated by the typical online learning evaluation method, test-then-train (prequential) [13], on a single epoch. Each individual example is used to test the model before it is used for training. Prequential approach enables the algorithm to make maximum usage of the available data, and does not require any hold-out training set in essence [37]. It could be found from [7,13] that the prequential error S is computed by the accumulated sum of the loss function:

$$S = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{test}), \tag{24}$$

[2] https://sci2s.ugr.es/keel/datasets.php.
[3] http://archive.ics.uci.edu/ml/.

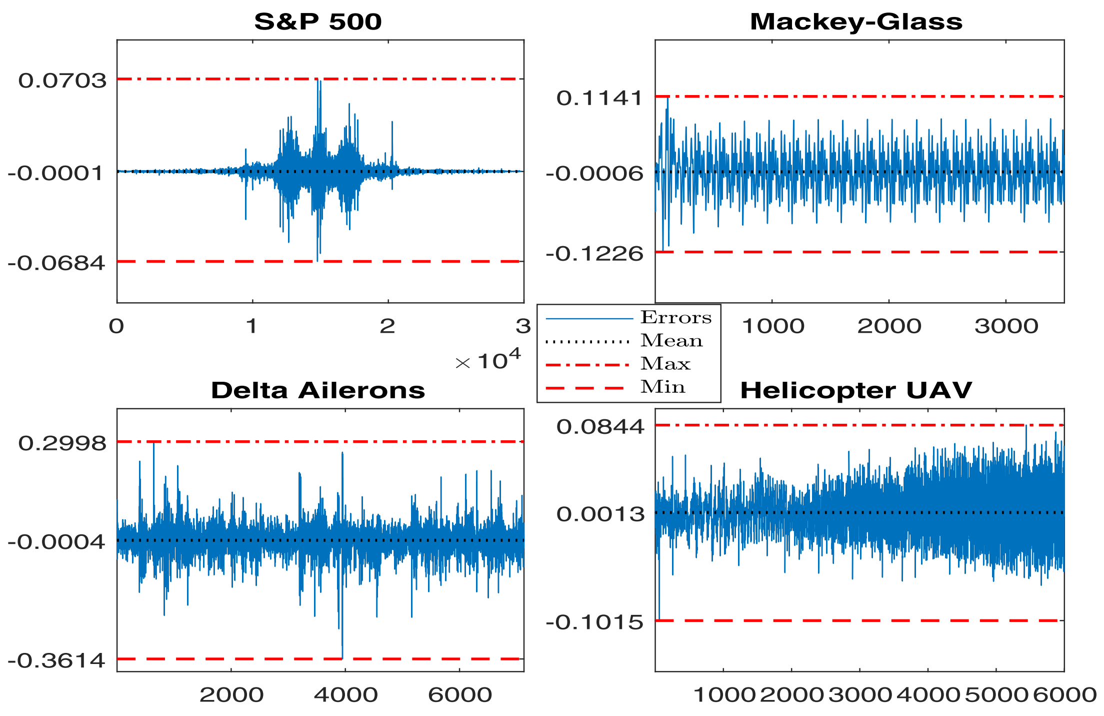

Fig. 3. Online testing errors.

Table 1
Example 1: S&P 500 daily closing prices.

	No. of rules (AVG.)	NDEI
GENEFIS [40]	2	0.07
PANFIS [38]	4	0.09
eT2RFNN [41]	2	0.04
eTS [3]	14	0.04
Simpl_eTS [2]	7	0.04
ANFIS [16]	32	0.02
LEOA[14]	52	0.1229
SEFS[15]	2(1.2835)	0.0182
EFS-SLAT $\lambda = 0.8$	31(19.6604)	0.0156
EFS-SLAT $\lambda = 0.85$	30(20.0015)	0.0156
EFS-SLAT $\lambda = 0.9$	27(17.3987)	0.0156
EFS-SLAT $\lambda = 0.95$	25(13.8840)	0.0156
EFS-SLAT $\lambda = 0.99$	**23(10.7570)**	**0.0156**

where $L(y_i, \hat{y}_i^{test})$ is the loss function $L(y_i, \hat{y}_i^{test}) = (y_i - \hat{y}_i^{test})^2$. For all these examples, EFS-SLAT has been operated on an online manner. Not only the testing accuracy computed based on the testing set, but also the online testing RMSEs, NDEIs with regard to all the data in each dataset, as well as the prequential errors are presented.

Additionally, the full trajectories of the online learning errors and thresholds ϵ_n^a and ϵ_n^m with regard to the best results (for Example 1–4) are presented in the Figs. 3 and 4, respectively.

4.1. Example 1: S&P 500 closing price prediction

The daily closing prices of S&P500 are collected from the Yahoo! Finance website, ranging from 03.01.1950 to 12.03.2009 (60 years). There are 14,893 data points in total. We use EFS-SLAT to make online predictions for the original time series and its flipped time series with 29,786 data points. The first half of the data are used for training and the remaining data are used for testing. The regression model is (25):

$$\hat{x}(t+1) = f(x(t-4), x(t-3), x(t-2), x(t-1), x(t)). \quad (25)$$

The comparison results are shown in Table 1 which indicates that EFS-SLAT achieves precious prediction results. Moreover, the online testing RMSE is 0.0045, NDEI is 0.0157 for all the values of λ except 0.8. When $\lambda = 0.8$, the NDEI equals to 0.0156. For λ equals to [0.8, 0.85, 0.9, 0.95, 0.99], the prequential error S is about [0.5972, 0.5995, 0.5994, 0.6035, 0.6003].

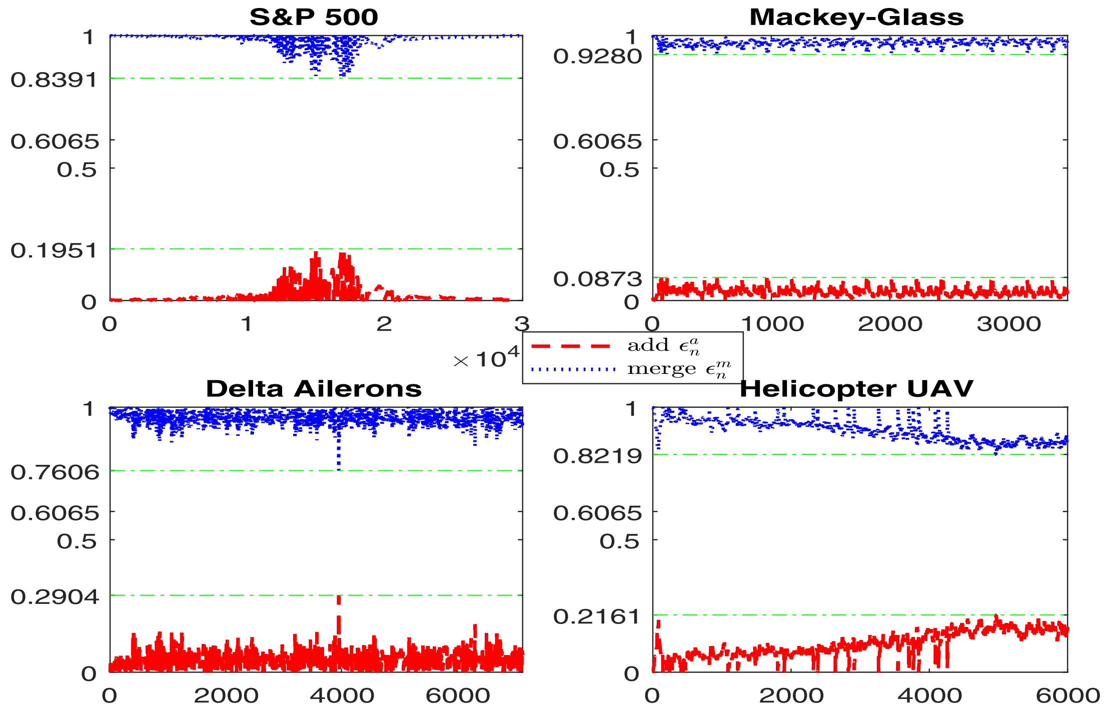

Fig. 4. Trajectories of the thresholds.

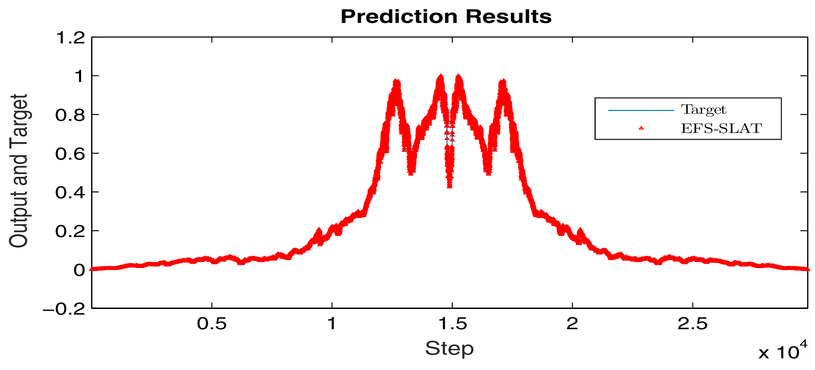

Fig. 5. Fitting results for *Example 1*.

Although EFS-SLAT uses more rules than some of the other approaches, the EFS-SLAT can achieve more accurate results than all of the others, no matter which forgetting factor is used and without requirement for setting the thresholds. In this sense, EFS-SLAT is still a preferable approach, as the high accuracy is the priority goal in system learning and identification. Based on the prequential error, EFS-SLAT achieves the most accurate result when $\lambda = 0.8$; whereas the same testing results as $\lambda = [0.8, 0.85, 0.9, 0.95]$ could be obtained with a smaller number of rules when $\lambda = 0.99$. Additionally, observing from Fig. 5, Figs. 3 and 4 it can be concluded that the thresholds have larger fluctuations when time series have more significant nonstationary phenomenon than the smooth areas.

4.2. Example 2: Mackey–Glass chaotic time series prediction

Mackey–Glass Chaotic time series is generated from (26):

$$\frac{d[x(t)]}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t), \tag{26}$$

in which $\tau = 17$, $x(0) = 1.2$. The total number of observations generated is 6000, 3000 of which are taken from $201 \leq t \leq 3200$, and 500 data points are chosen from $5001 \leq t \leq 5500$. These 3000 and 500 data are used for online training

Table 2
Example 2: Mackey–Glass chaotic time series.

	No. of rules (AVG.)	NDEI
DENFIS [17]	58	0.278
GENEFIS(C) [40]	19	0.280
PANFIS [38]	19	0.301
eT2RFNN [41]	3	0.32
GSETSK [35]	19	0.347
eTS [3]	9	0.3805
Simpl_eTS [2]	11	0.3940
SAFIS [42]	6	0.3760
SPLAFIS [36]	30	0.279
DeTS [6]	3	0.440
GEFNS [5]	5	0.2635
eMG($\Sigma_{init} = 10^{-3}I_4, w = 20$) [18]	58	0.139
LEOA [14]	42	0.2480
SEFS [15]	4(1.3043)	0.1287
EFS-SLAT $\lambda = 0.8$	**8(5.2529)**	**0.1140**
EFS-SLAT $\lambda = 0.85$	7(4.8231)	0.1160
EFS-SLAT $\lambda = 0.9$	6(3.8314)	0.1251
EFS-SLAT $\lambda = 0.95$	8(3.0637)	0.1364
EFS-SLAT $\lambda = 0.99$	8(2.5749)	0.1384

Table 3
Example 2: Online testing results for all the data.

λ	0.8	0.85	0.9	0.95	0.99
RMSE	**0.0314**	0.0319	0.0327	0.0350	0.0347
NDEI	**0.1411**	0.1435	0.1470	0.1575	0.1561
S	**3.4440**	3.5612	3.7409	4.2929	4.2147

Table 4
Example 3: Delta Ailerons data.

	No. of rules (AVG.)	RMSE
SAFIS [42]	14	0.0549
eTS [3]	4	0.0513
Simpl_eTS [2]	4	0.0512
GEFNS [5]	3	0.0502
SEFS [15]	7(1.5112)	0.0476
EFS-SLAT $\lambda = 0.8$	5(1.8174)	0.0413
EFS-SLAT $\lambda = 0.85$	5(1.7312)	0.0413
EFS-SLAT $\lambda = 0.9$	**4(1.6287)**	**0.0412**
EFS-SLAT $\lambda = 0.95$	4(1.5879)	0.0413
EFS-SLAT $\lambda = 0.99$	4(1.5932)	0.0412

and testing, respectively. The prediction model has the form

$$\hat{x}(t+85) = f(x(t-18), x(t-12), x(t-6), x(t)). \tag{27}$$

Numerical results are displayed in Table 2.

Table 2 illustrates that EFS-SLAT with $\lambda = 0.8$ achieves the best results, and the results obtained by other values of the λ are also comparable with the state-of-the-art approaches. The online learning results with regard to all the data points are presented in the Table 3. The best results are obtained by $\lambda = 0.8$ regarding to RMSE, NDEI and the prequential errors. Fig. 3 shows that the online testing accuracy for the whole data set varies between -0.1226 and 0.1141; Fig. 4 also presents the full trajectories for ϵ_n^a and ϵ_n^m. It can be observed that the largest value of ϵ_n^a and the smallest value of ϵ_n^m are 0.0873 and 0.9280, respectively.

4.3. Example 3: Delta Ailerons data

Delta Ailerons data from KEEL-dataset are applied in this example. They are obtained from the task of controlling the ailerons of a F16 aircraft. Five inputs named as “RollRate”, “PitchRate”, “currPitch”, “currRoll” and “diffRollRate” are applied to predict the output “Sa”. The same as [5], we use the first 3000 points for training, and the remaining 4129 points for evaluation. Table 4 indicates that, no matter which forgetting factor is selected, the EFS-SLAT can give better results compared to other state-of-the-art approaches. Besides, $\lambda = 0.9$ gives the most accurate results observed from both Tables 4 and 5. The trajectories of the errors, ϵ_n^a and ϵ_n^m are shown in the Figs. 3 and 4, respectively.

Table 5
Example 3: Online testing results for all the data.

λ	0.8	0.85	0.9	0.95	0.99
RMSE	0.0408	0.0408	**0.0407**	0.0407	0.0407
NDEI	0.5793	0.5791	**0.5776**	0.5785	0.5779
S	11.8692	11.8598	**11.7998**	11.8351	11.8118

Table 6
Example 4: Helicopter unmanned aerial vehicle streaming data.

	No. of rules (AVG.)	RMSE	NDEI
eTS [3]	3	0.0535	0.8352
Simpl_eTS [2]	3	0.0534	0.8336
GENEFIS [40]	2	0.0355	0.5541
PANFIS [38]	9	0.0362	0.5652
Type-1 PALM(L) [12]	6	0.0363	0.5668
Type-1 PALM(G) [12]	11	0.0313	0.4886
EFS-SLAT $\lambda = 0.8$	9(5.8460)	0.0311	0.4846
EFS-SLAT $\lambda = 0.85$	8(5.4463)	0.0312	0.4862
EFS-SLAT $\lambda = 0.9$	7(4.7357)	0.0309	0.4820
EFS-SLAT $\lambda = 0.95$	**11(4.2770)**	**0.0305**	**0.4758**
EFS-SLAT $\lambda = 0.99$	9(2.0945)	0.0316	0.4920

Table 7
Example 4: Online testing results for all the data.

λ	0.8	0.85	0.9	0.95	0.99
RMSE	0.0252	0.0253	0.0252	0.0245	0.0247
NDEI	0.2030	0.2035	0.2024	0.1975	0.1991
S	3.8184	3.8367	3.7961	3.6138	3.6749

Table 8
Example 5: MPG and Boston Housing data.

	MPG			Boston Housing		
	MAE $\pm$ STD	max MAE	rule	MAE $\pm$ STD	max MAE	rule
genfis2 [48]	2.23 $\pm$ 0.85	3.88	6	3.14 $\pm$ 1.31	6.53	4
ANFIS [16]	2.41 $\pm$ 0.84	4.07	16	3.59 $\pm$ 1.39	6.56	4
SparseFIS uncon. [28]	2.14 $\pm$ 0.78	4.08	22	3.60 $\pm$ 1.56	6.86	19
FLEXFIS [32]	2.17 $\pm$ 0.73	3.59	11	2.98 $\pm$ 1.27	6.09	6
Gen-Smart-EFS [26]	2.09 $\pm$ 0.62	3.60	5.8	2.94 $\pm$ 1.18	5.84	4.9
EFS-SLAT$\lambda = 0.8$	2.097 $\pm$ 0.393	2.836	5.306	**2.817** $\pm$ **0.427**	**3.339**	**6.190**
EFS-SLAT$\lambda = 0.85$	2.102 $\pm$ 0.316	2.578	5.322	2.842 $\pm$ 0.551	3.596	6.287
EFS-SLAT$\lambda = 0.9$	2.092 $\pm$ 0.395	2.866	5.465	2.843 $\pm$ 0.680	3.827	6.228
EFS-SLAT$\lambda = 0.95$	2.102 $\pm$ 0.259	2.556	5.491	2.870 $\pm$ 0.476	3.498	6.188
EFS-SLAT$\lambda = 0.99$	**2.088** $\pm$ **0.195**	**2.386**	**5.564**	2.867 $\pm$ 0.527	3.809	6.121

4.4. Example 4: Helicopter unmanned aerial vehicle streaming data

The Rotary Unmanned Aerial Vehicles (RUAV) dataset is collected from an Align Trex450 Pro-Direct Flight Control (DFC), fly bar-less, helicopter made in Taiwan. We use the same dataset conducted by the UAV laboratory of the UNSW Canberra campus, as [12]. The dataset is formed by 6000 samples, which are highly nonlinear and nonstationary. The first 3600 samples are used for training, and the remainder are used for testing. The input and output are set the same as [12]. Comparison results and the online learning accuracies are presented in the Tables 6 and 7, respectively. EFS-SLAT with $\lambda = 0.95$ achieves the best result.

4.5. Example 5: Data sets from UCI repository

MPG and Boston Housing data taken from UCI repository are used. Following on [26] and [27], the evaluation procedure follows a 10-fold cross-validation. The results 'MAE' of EFS-SLAT in Table 8 are the best results based on 1000 Monte Carlo CV procedure; 'STD' and the 'rule' are the corresponding standard deviation towards the CV folds and the average rule numbers, respectively; Besides, 'max MAE' is the corresponding maximum error towards all the data (the worst-case prediction error) [27]. The results of other methods are copied from [26], and they are picked as their best results. It can be seen from Table 8 that EFS-SLAT gives either higher accuracy, or smaller standard deviation and maximum MAE with a small amount of fuzzy rules.

5. Conclusions

In order to overcome the weakness of the existing EFSs where thresholds are treated as the predefined parameters and the selection and determination of thresholds are completely relied on the trial and error strategy, we propose an EFS online learning method — EFS-SLAT which uses a simple and easily understood structure to enhance the flexibility and effectiveness of the learning. EFS-SLAT automatically identifies the thresholds to control the structure evolving modules which refer to rule generation, merging and pruning. These thresholds are evolved by automatically making use of the overfitting and underfitting information, which is accomplished through building the thresholds as functions of the cumulative absolute training error. Furthermore, thresholds for structural evolving are changing dynamically in order to cooperate with, and to restrict, each other. In this way, the possible overfitting and underfitting effect, accompanied with the fast rule generation speed and the rule merging speed could be effectively balanced. Moreover, it decides whether a structure evolving module should be carried out by comparing the online training errors, in order to avoid unnecessary structure change. Numerical experiments confirm that EFS-SLAT has outstanding prediction accuracies compared with many state-of-the-art approaches.

Our future research work will focus on identifying and developing other indicators which can give a more accurate description of the overfitting, underfitting, data nonlinearity and nonstationary, in order to complement the information contained in the training errors; and better reinforce the algorithm. Besides, the EFS with short memory of the historical information will be investigated in order to better assist the system in the structure evolving.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (no. 91746103). The authors would like to thank the reviewers, the Associate Editor and the Editor-in-Chief for their constructive and very helpful comments. The authors would also like to thank Ms. Rosemary Goodier from the department of Strategic Marketing and Communications, University of Manchester, for proof reading.

References

[1] P. Angelov, Fuzzily connected multimodel systems evolving autonomously from data streams, IEEE Trans. Syst. Man Cybern.Part B 41 (4) (2011) 898–910.
[2] P. Angelov, D. Filev, Simpl_ets: a simplified method for learning evolving Takagi–Sugeno fuzzy models, in: Fuzzy Systems, 2005. FUZZ'05. The 14th IEEE International Conference on, IEEE, 2005, pp. 1068–1073.
[3] P.P. Angelov, D.P. Filev, An approach to online identification of Takagi–Sugenofuzzy models, IEEE Trans. Syst. Man Cybern.Part B 34 (1) (2004) 484–498.
[4] A. Ashfahani, M. Pratama, Autonomous deep learning: continual learning approach for dynamic environments, arXiv:1810.07348 (2018).
[5] R.-J. Bao, H.-J. Rong, P.P. Angelov, B. Chen, P.K. Wong, Correntropy-based evolving fuzzy neural system, IEEE Trans. Fuzzy Syst. 26 (3) (2018) 1324–1338.
[6] R.D. Baruah, P. Angelov, Dec: dynamically evolving clustering and its application to structure identification of evolving fuzzy models, IEEE Trans. Cybern. 44 (9) (2014) 1619–1631.
[7] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, B. Pfahringer, Efficient online evaluation of big data stream classifiers, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, 2015, pp. 59–68.
[8] M.-Y. Chen, D.A. Linkens, Rule-base self-generation and simplification for data-driven fuzzy models, Fuzzy SetsSyst. 142 (2) (2004) 243–265.
[9] J. de Jesús Rubio, Sofmls: online self-organizing fuzzy modified least-squares network, IEEE Trans. Fuzzy Syst. 17 (6) (2009) 1296–1309.
[10] J. de Jesús Rubio, Error convergence analysis of the sufin and csufin, Appl. Soft Comput. 72 (2018) 587–595.
[11] J. de Jesus Rubio, E. Lughofer, J.A. Meda-Campaña, L.A. Páramo, J.F. Novoa, J. Pacheco, Neural network updating via argument kalman filter for modeling of Takagi–Sugeno fuzzy models, J. Intell. Fuzzy Syst. 35 (2) (2018) 2585–2596.
[12] M.M. Ferdaus, M. Pratama, S. Anavatti, M.A. Garratt, Palm: an incremental construction of hyperplanes for data stream regression, IEEE Trans. Fuzzy Syst.(Early Access) (2019), doi:10.1109/TFUZZ.2019.2893565.
[13] J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall/CRC, 2010.
[14] D. Ge, X.-J. Zeng, Learning evolving t–s fuzzy systems with both local and global accuracy–a local online optimization approach, Appl. Soft Comput. 68 (2018) 795–810.
[15] D. Ge, X.-J. Zeng, A self-evolving fuzzy system which learns dynamic threshold parameter by itself, IEEE Trans. Fuzzy Syst. (Early Access) (2018), doi:10.1109/TFUZZ.2018.2886154.
[16] J.-S. Jang, Anfis: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. ManCybern. 23 (3) (1993) 665–685.
[17] N.K. Kasabov, Q. Song, Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 144–154.
[18] A. Lemos, W. Caminhas, F. Gomide, Multivariable Gaussian evolving fuzzy modeling system, IEEE Trans. Fuzzy Syst. 19 (1) (2011) 91–104.
[19] G. Leng, T.M. McGinnity, G. Prasad, An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network, Fuzzy SetsSyst. 150 (2) (2005) 211–243.
[20] X. Li, H. Li, B. Sun, F. Wang, Assessing information security risk for an evolving smart city based on fuzzy and grey fmea, J. Intell. Fuzzy Syst. 34 (4) (2018) 2491–2501.
[21] Y. Liu, Z. Wang, Y. Yuan, F.E. Alsaadi, Partial-nodes-based state estimation for complex networks with unbounded distributed delays, IEEE Trans. Neural Netw. Learn.Syst. 29 (8) (2018) 3906–3912.
[22] E. Lughofer, On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues, Inf. Sci. 251 (2013) 22–46.
[23] E. Lughofer, Evolving Fuzzy Systems–fundamentals, Reliability, Interpretability, Useability, Applications, in: Handbook on Computational Intelligence: Volume 1: Fuzzy Logic, Systems, Artificial Neural Networks, and Learning Systems, World Scientific, 2016, pp. 67–135.
[24] E. Lughofer, On-line active learning: a new paradigm to improve practical useability of data stream modeling methods, Inf. Sci. 415 (2017) 356–376.

[25] E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Appl. Soft Comput. 11 (2) (2011) 2057–2068.
[26] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, Evol. Syst. 6 (4) (2015) 269–292.
[27] E. Lughofer, S. Kindermann, Rule weight optimization and feature selection in fuzzy systems with sparsity contraints, in: IFSA/EUSFLAT Conf., Citeseer, 2009, pp. 950–956.
[28] E. Lughofer, S. Kindermann, Sparsefis: data-driven learning of fuzzy systems with sparsity constraints, IEEE Trans. Fuzzy Syst. 18 (2) (2010) 396–411.
[29] E. Lughofer, M. Pratama, Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models, IEEE Trans. Fuzzy Syst. 26 (1) (2018) 292–309.
[30] E. Lughofer, M. Pratama, I. Skrjanc, Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 1854–1865.
[31] E. Lughofer, M. Sayed-Mouchaweh, Autonomous data stream clustering implementing split-and-merge concepts–towards a plug-and-play approach, Inf. Sci. 304 (2015) 54–79.
[32] E.D. Lughofer, Flexfis: a robust incremental learning approach for evolving takagi–sugeno fuzzy models, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1393–1410.
[33] L. Maciel, R. Ballini, F. Gomide, An evolving possibilistic fuzzy modeling approach for value-at-risk estimation, Appl. Soft Comput. 60 (2017) 820–830.
[34] L. Maciel, F. Gomide, R. Ballini, Enhanced evolving participatory learning fuzzy modeling: an application for asset returns volatility forecasting, Evol. Syst. 5 (2) (2014) 75–88.
[35] N.N. Nguyen, W.J. Zhou, C. Quek, Gsetsk: a generic self-evolving tsk fuzzy neural network with a novel Hebbian-based rule reduction approach, Appl. Soft Comput. 35 (2015) 29–42.
[36] R.J. Oentaryo, M.J. Er, S. Linn, X. Li, Online probabilistic learning for fuzzy inference system, Expert Syst. Appl. 41 (11) (2014) 5082–5096.
[37] A. Patil, V. Attar, Framework for performance comparison of classifiers, in: Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20–22, 2011, Springer, 2012, pp. 681–689.
[38] M. Pratama, S.G. Anavatti, P.P. Angelov, E. Lughofer, Panfis: a novel incremental learning machine, IEEE Trans. Neural Netw. Learn.Syst. 25 (1) (2014) 55–68.
[39] M. Pratama, S.G. Anavatti, M. Joo, E.D. Lughofer, Pclass: an effective classifier for streaming examples, IEEE Trans. Fuzzy Syst. 23 (2) (2015) 369–386.
[40] M. Pratama, S.G. Anavatti, E. Lughofer, Genefis: toward an effective localist network, IEEE Trans. Fuzzy Syst. 22 (3) (2014) 547–562.
[41] M. Pratama, J. Lu, E. Lughofer, G. Zhang, M.J. Er, An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks, IEEE Trans. Fuzzy Syst. 25 (5) (2017) 1175–1192.
[42] H.-J. Rong, N. Sundararajan, G.-B. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction, Fuzzy SetsSyst. 157 (9) (2006) 1260–1275.
[43] M. Setnes, R. Babuska, U. Kaymak, H.R. van Nauta Lemke, Similarity measures in fuzzy rule base simplification, IEEE Trans. Syst. Man Cybern.Part B 28 (3) (1998) 376–386.
[44] A.M. Silva, W. Caminhas, A. Lemos, F. Gomide, A fast learning algorithm for evolving neofuzzy neuron, Appl. Soft Comput. 14 (2014) 194–209.
[45] A.M. Soares, B.J. Fernandes, C.J. Bastos-Filho, Pyramidal neural networks with evolved variable receptive fields, Neural Comput. Appl. 29 (12) (2018) 1443–1453.
[46] H. Trevor, T. Robert, F. JH, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, New York, NY: Springer, 2009.
[47] S.W. Tung, C. Quek, C. Guan, Et2fis: an evolving type-2 neural fuzzy inference system, Inf. Sci. 220 (2013) 124–148.
[48] R.R. Yager, D.P. Filev, Approximate clustering via the mountain method, IEEE Trans. Syst. Man Cybern. 24 (8) (1994) 1279–1284.