Information Sciences 579 (2021) 623–647

Contents lists available at ScienceDirect

Information Sciences

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

A self-adaptive fuzzy learning system for streaming data prediction

Xiaowei Gu [a,*], Qiang Shen [a]

[a] Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, UK

ARTICLE INFO

Article history:
Received 10 March 2021
Received in revised form 4 August 2021
Accepted 7 August 2021
Available online 10 August 2021

Keywords:
Evolving fuzzy system
Fuzzy inference
Streaming data
Stability

ABSTRACT

In this paper, a novel self-adaptive fuzzy learning (SAFL) system is proposed for streaming data prediction. SAFL self-learns from data streams a predictive model composed of a set of prototype-based fuzzy rules, with each of which representing a certain local data distribution, and continuously self-evolves to follow the changing data patterns in non-stationary environments. Unlike conventional evolving fuzzy systems, both the fuzzy inference and consequent parameter learning schemes utilised by SAFL are simplified so that only a small number of selected fuzzy rules within the rule base are involved in system output generation and parameter updating during a learning cycle. Such simplification not only significantly reduces the system's computational complexity but also increases its prediction precision. In addition, both theoretical and empirical investigations guarantee the stability of the resulting SAFL. Comparative experimental studies on a wide variety of benchmark and real-world problems demonstrate that SAFL is able to learn from streaming data in a highly efficient manner and to make predictions with a great accuracy, revealing the effectiveness and validity of the proposed approach.

1. Introduction

Real-time streaming data processing is a key to decision-making in time-critical applications, such as high-frequency trading, autonomous driving, video surveillance, etc. However, non-stationary data stream processing and modelling remains a challenging task for computational intelligence models due to the problems of continuously arriving new samples and dynamically changing underlying attributes [42]. To effectively handle data streams with such characteristics, models learning to capture the essence of such data need to be equipped with a flexible system structure entailing an incremental learning capability. This is in order to support self-adaptation or self-evolution to survive the changing environments. Particularly, the associated learning algorithms are expected to be able to learn how to process new data in an efficient, and ideally, "one pass" manner under restricted time and memory bound in the absence of historical data [7].

Deep neural networks (DNNs) have achieved excellent results on a varity of real-world problems, such as natural language processing and image classification [26]. DNNs are capable of automatically learning multiple levels of representations from raw data through a general-purpose learning procedure and usually offer high predictive precision on large-scale, complex problems. However, conventional DNNs are "black box" models with a large number of hyper-parameters to learn. Their training process is computationally expensive and restricted to offline, while requiring full retraining if data with unfamiliar patterns is observed. As a result, it is generally very difficult to apply DNNs for non-stationary data stream processing.

* Corresponding author.
E-mail address: xig4@aber.ac.uk (X. Gu).

https://doi.org/10.1016/j.ins.2021.08.023
0020-0255/

Recently, a number of more advanced DNN models based on incremental and continual learning have been proposed for streaming data processing, such as PNN [39], DEN [48], ADL [5], NADINE [32], MUSE-RNN [7]. Nevertheless, such models remain to have limited model transparency and their inferences have little interpretability, while lacking theoretical proof of stability and convergence.

Evolving fuzzy systems (EFSs) are a powerful tool widely used for real-time non-stationary problem approximation. In contrast with DNNs, EFSs are designed to simultaneously self-develop and self-update both the system structure and the meta–parameters online from data streams, in an effort to capture and reflect the dynamical changes of data patterns. With a highly transparent system structure and human-understandable fuzzy reasoning mechanism, EFSs provide an effective solution toward explainable artificial intelligence (xAI) [16]. Indeed, EFSs have received great attention since the underlying concept was firstly introduced at the beginning of this century [1,23].

The system structure of EFSs is usually based on IF…THEN…fuzzy production rules. In general, EFSs may employ different structural evolving schemes for fuzzy rule identification and premise parameter learning [17,34]. The key idea of such evolving schemes is to group streaming data into clusters utilising recursive clustering techniques and associate each individual cluster to a particular fuzzy rule [42]. They enable EFSs to continuously self-learn from new data samples "on the fly" in an exploratory manner, efficiently transforming the learned knowledge into human-interpretable fuzzy rules. For example, one of the earliest EFSs named evolving Takagi–Sugeno (eTS) [1] uses the eClustering algorithm to construct a set of first-order fuzzy rules from data streams. Indeed, there have been many similar approaches proposed in the literature, and their differences mostly lie in the way that recursive clustering is performed [3,36]. Depending on which structural evolving scheme is adopted, different EFSs may use different criteria for rule evolution, merging, pruning and splitting [42]. They each have their own pros and cons however, leading to a diverse range of EFS behaviours toward different types of approach depending on the nature of the application problem at hand. Nonetheless, a vast majority of EFSs utilise recursive least squares (RLS)-based techniques for learning the model parameters [19,38]. Note that given the production representation of fuzzy systems, the model complexity in terms of the adaptive parameters is far simpler than that of DNNs.

Evolving neuro-fuzzy systems (ENFSs) are a popular alternative for data stream modelling closely related to EFSs. The structure of neuro-fuzzy models is composed of neurons, but fuzzy rules can be readily extracted if desired, at any time [42]. The majority of neuro-fuzzy models are based on radical basis function models or their generalised forms. The most representative and earliest ENFSs include DENFIS [23] and SOFNN [24]. More recently developed ones, e.g., PANFIS [29], further provide an upper bound for the overall approximation error, providing an equivalent theoretical proof of convergence and stability.

Many other successful EFSs and ENFSs have been developed over the last two decades, which include, but are not limited to, Simpl_eTS [2], SAFIS [36], eClass [4], FLEXFIS [27], McFIS[45], GENEFIS [30], ALMMo [3], CENFS [6], PALM [14], SEFS [18], eGAUSS+ [43] and HiPCA [10]. Nowadays, EFSs have been widely implemented for real-world applications concerning streaming data processing such as time series prediction and systems identification [3,9]. Interested readers are referred to the recent review paper [42] for more details about the latest developments in the area of EFSs and ENFSs as well as their applications.

Typically, given a particular input sample, the overall output of an EFS is formulated as the fuzzily weighted sum of the consequents of those individual fuzzy rules within the system [1]. Thus, fuzzy rules with prototypes that are spatially closer to the input sample usually contribute more in the overall output. However, involving all fuzzy rules within the rule base for system output generation does not necessarily help EFSs to perform preciser inferencing [25]. Yet, such an inferencing scheme generally leads to higher computational complexity. Also, EFSs often need to update the consequent parameters of all the fuzzy rules simultaneously based on the current input–output during each processing cycle [3], while expecting the system structure and parameters to self-adapt to the non-stationary environment rapidly. Nonetheless, updating the fuzzy rules in response to the current input sample that represents a very different data pattern is completely unnecessary considering the marginal membership values these rules assign to it. Besides, this type of parameter learning scheme may add extra computational burden to EFSs, especially when the dimensionality of data is high or the size of fuzzy rule base is very large. To improve the computational efficiency of EFSs, an agiler inference scheme is desirable, and the commonly used consequent parameter learning mechanism needs to be simplified.

Based on the above observations, and inspired by the underlying idea of using the least number of rules to perform approximate reasoning [25], a novel self-adaptive fuzzy learning (SAFL) fuzzy system is herein proposed for streaming data prediction. SAFL is able to self-organise a set of fuzzy rules representing the local models of data distribution to develop a predictive model with high precision. Its structural evolving scheme is driven by empirically observed data samples on the basis of their integral properties and is, therefore, highly objective. Comparing with alternative EFSs, both the fuzzy inference and consequent parameter learning schemes of the proposed system are simpler. During each learning cycle, only a small number of activated fuzzy rules by the current input sample are selected and involved in system output generation and consequent parameter updating. This simplification largely reduces the computational complexity of SAFL and further improves its prediction precision. Most importantly, it is justified through theoretical analysis that the stability of the SAFL system is guaranteed. Experimental studies conducted over a variety of benchmark and real-world datasets under the commonly used experimental protocols, including the prequential test-then-train method for data stream problems [15], demonstrate the efficacy of the proposed SAFL system.

Briefly, the main contributions of this paper include:

1. Highly flexible fuzzy inference and consequent parameter learning scheme with rule selection that jointly improve both computational efficiency and prediction precision of the resulting SAFL system; and

2. Theoretical analysis that guarantees the stability of the proposed approach, extensible to general EFSs that involve local consequent parameter learning via the RLS algorithm.

The remainder of this paper is organised as follows. Section 2 gives the problem statement. Technical details of SAFL are presented in Section 3, including an analysis of the approach's computational complexity. The learning stability is theoretically examined in Section 4. Systematic experimental results are provided in Section 5. This paper is concluded in Section 6.

2. Problem Statement

A conventional multiple inputs-single output (MISO) first-order EFS is composed of N first-order fuzzy rules defined in the following form ($n = 1, 2, \ldots, N$) [3]:

$$\mathrm{R}_n: \ \mathrm{IF}(x \sim p_n) \ \mathrm{THEN} \ (y_n = \bar{x}^T a_n); \tag{1}$$

where $x = [x_1, x_2, \ldots, x_L]^T \in \Re^L$ is the input vector of the fuzzy rules; $\Re^L$ is the L dimensional real data space; $\bar{x} = [1, x^T]^T$; y_n is the consequent or output of $\mathrm{R}_n, n = 1, \ldots, N$; $p_n = [p_{n,1}, p_{n,2}, \ldots, p_{n,M}]^T$ and $a_n = [a_{n,0}, a_{n,1}, a_{n,2}, \ldots, a_{n,M}]^T$ are the respective prototype and consequent parameters of R_n.

For a certain input sample, x_k, its firing strength to R_n is defined by the following Gaussian function [20]:

$$\mu_n(x_k) = e^{-\frac{\|x_k - p_n\|^2}{\sigma_n^2}}; \tag{2}$$

where σ_n is defined as the radius of the area of influence around p_n.

The final output of the system is defined as the fuzzily weighted sum of consequents of all N fuzzy rules within the rule base [20]:

$$y_k = f(x_k) = \sum_{n=1}^{N} \lambda_{n,k} y_{n,k} = \sum_{n=1}^{N} \lambda_{n,k} \bar{x}_k^T a_n; \tag{3}$$

where $\bar{x}_k^T = [1, x_k^T]^T$; λ_n is the normalised firing strength of R_n computed by:

$$\lambda_{n,k} = \frac{\mu_n(x_k)}{\sum_{j=1}^{N} \mu_j(x_k)}. \tag{4}$$

In the literature, most EFSs update the consequent parameters of all fuzzy rules within the candidate rule base, with respect to the input sample x_k, using weighted or fuzzily weighted RLS algorithms. Through adjusting the weights of the fuzzy rules based on the distances between the precompiled prototypes and the current sample, conventional EFSs are able to achieve a soft and smooth adaptation of model parameters. Considering the classical fuzzily weighted RLS algorithm [1], the consequent parameters of the fuzzy rules are updated individually by Eqns. (5) and (6):

$$\mathbf{\Theta}_n \leftarrow \mathbf{\Theta}_n - \frac{\lambda_{n,k} \mathbf{\Theta}_n \bar{x}_k \bar{x}_k^T \mathbf{\Theta}_n}{1 + \lambda_{n,k} \bar{x}_k^T \mathbf{\Theta}_n \bar{x}_k}; \tag{5}$$

$$a_n \leftarrow a_n - \lambda_{n,k} \mathbf{\Theta}_n \bar{x}_k (\bar{x}_k^T a_n - y_k); \tag{6}$$

where $n = 1, \ldots, N$; and a_n and $\mathbf{\Theta}_n$ are the consequent parameters and covariance matrix of R_n, respectively.

As aforementioned, the commonly used fuzzy inferencing and consequent parameter learning scheme of conventional EFSs involves all fuzzy rules within the rule base of the system in response to a given input sample. However, involving all fuzzy rules in fuzzy inferencing does not necessarily improve the system predictive precision because those fuzzy rules with marginal rule-firing strength often play a negligible role in deriving the final output. Correspondingly, updating the consequent parts of such fuzzy rules with the RLS algorithm only introduces tiny changes on the consequent parameters with no improvement in the overall system performance, but would significantly increase the computational complexity (mostly due to the updating of covariance matrices), especially for high-dimensional problems, and may lead to overfitting. Hence, to achieve greater predictive precision and lower computational efficiency, a more flexible fuzzy inference and consequent parameter learning scheme is needed. For clarify, the key notations used in this paper and their definitions are summarised in Table 1.

3. SAFL system

3.1. System architecture

The general structure of SAFL is presented in Fig. 1, comprising N first-order prototype-based fuzzy rules in the same form of Eqn. (1). Note that unlike the vast majority of EFSs in the literature, such as DENFIS [23], eTS [1], SAFIS [36] and their successors, which use cluster means as the prototypes of fuzzy rules, the prototypes of SAFL are data samples directly selected

Table 1
List of Key Notations and the Definitions.

Notation	Definition
x	Input sample
k	Current time instance
y	Reference output corresponding to x
$\hat{y}$	System output corresponding to x
N	Number of fuzzy rules
R_n	The n^{th} fuzzy rule
μ_n	Firing strength of R_n
λ_n	Normalised firing strength of R_n
p_n	Prototype/antecedent part of R_n
a_n	Consequent part of R_n
$\mathbf{\Theta}_n$	Covariance matrix of R_n
$\boldsymbol{\upsilon}$	Mean of observed data samples
χ	Average squared Euclidean norm of observed samples
$\mathbf{C}_n$	Cluster associated with R_n
c_n	Mean/centre of $\mathbf{C}_n$
S_n	Cardinality of $\mathbf{C}_n$
X_n	Average squared Euclidean norm of samples in $\mathbf{C}_n$
I_n	Time instance at which $\mathbf{C}_n$ is initialised
M_n	Average firing strength from I_n to current time instance
E_0	Inherent system approximation error
ϵ_n	Inherent approximation error of R_n
e	System prediction error
ϵ_0	Overall system inherent approximation error
τ_0	Upper limit of ϵ_0

from those empirically observed. These samples are assumed to be highly descriptive in the problem space with their original semantics unaltered. In contrast, clusters employed in classical EFSs are mathematical transformations of the original data that generally do not exist in the real world and hence, have no physical meaning. Therefore, SAFL can provide more intuitive information of the underlying problem domain than its peers.

Instead of involving all fuzzy rules to provide the overall system output like conventional EFSs (see Eqn. (3)), given a particular input vector, x, the output of SAFL is defined as the weighted sum of fuzzy outputs by a smaller number of more activated fuzzy rules within the system:

$$y = f(x) = \sum_{n=1}^{N^*} \lambda_n^* y_n^* = \sum_{n=1}^{N^*} \lambda_n^* \bar{x}^T a_n^*; \tag{7}$$

where N^* is the number of activated fuzzy rules by x, $1 \leqslant N^* \leqslant N$; y_n^* is the fuzzy output by the n^{th} activated fuzzy rule, R_n^*; λ_n^* is the corresponding normalised firing strength of R_n^*; and a_n^* is the vector of consequent parameters in R_n^*.

3.2. Autonomous learning scheme

The online autonomous learning process of SAFL consists of five distinct stages as detailed below.

Stage 1. System Initialisation.

SAFL is initiated by the first observed streaming sample, x_k ($k = 1$). Following the common practice in the literature [3,20], global meta–parameters of the system are set as below:

$$\boldsymbol{\upsilon} \leftarrow x_k; \chi \leftarrow ||x_k||^2; \tag{8}$$

where $\boldsymbol{\upsilon}$ and χ are the respective mean and average squared Euclidean norm of all data samples observed so far.

The first cluster, which represents one of the component models (or candidate rules) reflecting locality of the data distribution, is initialised by x_k as follows ($N \leftarrow 1$).

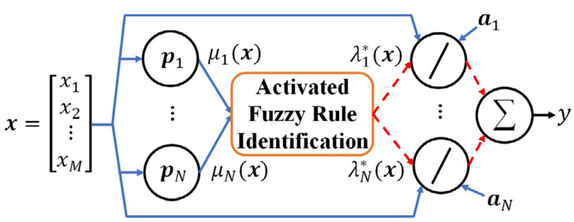


Fig. 1. Architecture of SAFL.

$$\mathbf{C}_N \leftarrow \{x_k\}; p_{N,k} \leftarrow x_k; S_{N,k} \leftarrow 1;$$
$$c_{N,k} \leftarrow x_k; X_{N,k} \leftarrow ||x_k||^2; I_N \leftarrow k; M_{N,k} \leftarrow 1; \tag{9}$$

where $p_{N,k}$ is the prototype of $\mathbf{C}_N$, which is a real data sample; $c_{N,k}$ is the mean of data samples associated with $\mathbf{C}_N$ at the current k^{th} time instance, $c_{N,k} = \frac{1}{S_{N,k}}\sum_{x \in \mathbf{C}_N} x$; $X_{N,k}$ is the corresponding average Euclidean norm of all data samples that are affiliated to $\mathbf{C}_N$, $X_{N,k} = \frac{1}{S_{N,k}}\sum_{x \in \mathbf{C}_N}||x||^2$; $S_{N,k}$ is the cardinality of $\mathbf{C}_N$; I_N denotes the time instance at which $\mathbf{C}_N$ is initialised; and $M_{N,k}$ is the average firing strength starting from I_N to the current time instance.

The first fuzzy rule, R_N is initialised in the same form as Eqn. (1) with $p_{N,k}$ as its premise part. Its consequent parameters, $a_{N,k}$ and the corresponding covariance matrix, $\boldsymbol{\Theta}_{N,k}$ are initialised by the following (as per [3,20]):

$$a_{N,k} \leftarrow 0_{(L+1)\times 1}; \boldsymbol{\Theta}_{N,k} \leftarrow \Omega_0 \mathrm{I}_{(L+1)\times(L+1)}; \tag{10}$$

where I is an $(L+1)\times(L+1)$ dimensional identity matrix; Ω_0 is an externally controlled parameter standard for covariance matrix initialisation used by the RLS algorithms [1,3,46]. In this work, $\Omega_0 = 1000$ unless otherwise stated (in the experimental studies).

Stage 2. System Output Generation.

Given a new input data sample, x_k ($k \leftarrow k+1$), the global meta–parameters, $\boldsymbol{\upsilon}$ and χ are updated by the following (again, as per [3,20]).

$$\boldsymbol{\upsilon} \leftarrow \boldsymbol{\upsilon} + \frac{x_k - \boldsymbol{\upsilon}}{k}; \chi \leftarrow \chi + \frac{||x_k||^2 - \chi}{k}. \tag{11}$$

The firing strength of x_k to each individual fuzzy rule, R_n $(n = 1, 2, \ldots, N)$ is then calculated by

$$\mu_n(x_k) = e^{-\frac{||x_k - p_{n,k}||^2}{\sigma_{n,k}^2}}; \tag{12}$$

where $\sigma_{n,k} = \sqrt{(\frac{\chi - ||\boldsymbol{\upsilon}||^2 + X_{n,k} - ||c_{n,k}||^2}{2})}$, which defines the radius of area influenced by $p_{n,k}$.

Note that $2(\chi - ||\boldsymbol{\upsilon}||^2)$ is the average squared Euclidean distance between any two data samples observed, and $2(X_{n,k} - ||c_{n,k}||^2)$ is the average squared Euclidean distance between any two data samples affiliated to $\mathbf{C}_n$ [3]. Thus, $\sigma_{n,k}^2$ can be defined as the average of $\chi - ||\boldsymbol{\upsilon}||^2$ and $X_{n,k} - ||c_{n,k}||^2$ to take into consideration both global and integral local properties of data. From this, the obtained firing strengths, $\mu_1(x_k), \mu_2(x_k), \ldots, \mu_N(x_k)$ are ranked in descending order, re-denoted as: $\mu_1^*(x_k), \mu_2^*(x_k), \ldots, \mu_N^*(x_k)$ $(\mu_1^*(x_k) \geqslant \mu_2^*(x_k) \geqslant \ldots \geqslant \mu_N^*(x_k))$.

Given the above, the top N_k^* activated fuzzy rules, $\mathrm{R}_1^*, \mathrm{R}_2^*, \ldots, \mathrm{R}_{N_k^*}^*$ can then be identified by

$$N_k^* = \mathrm{argmin}_{n=1,2,\ldots,N}(\sum_{i=1}^{n}\mu_i^*(x_k) > \gamma_0 \sum_{i=1}^{N}\mu_i(x_k)), \tag{13}$$

where $0 \leqslant \gamma_0 \leq 1$ and there is always $N_k^* \leqslant \gamma_0 N$. Clearly, if $\gamma_0 = 1$, all the fuzzy rules in the candidate rule base will be used to compute the system output. On the contrary, only the fuzzy rule(s) giving the highest firing strength will be used for producing the overall output if $\gamma_0 = 0$. In implementation, the recommended value of γ_0 is 0.5 unless otherwise stated (or specific domain knowledge is available to determine its value).

The selected rules, namely, $\mathrm{R}_1^*, \mathrm{R}_2^*, \ldots, \mathrm{R}_{N_k^*}^*$ are activated by the current input to produce higher firing strength values than the rest in the rule base. This is because x_k is spatially closer to $p_1^*, p_2^*, \ldots, p_{N_k^*}^*$, thereby better fitting the local distribution that is reflected by these rules. Note that instead of using a hardwired crisp threshold to select such more activated fuzzy rules, which is far less robust and goes against the intuition of utilising fuzzy representation, a self-adaptive rule selection mechanism is adopted in Eqn. (13), based on the ratio between the firing strengths of the selected fuzzy rules and that of the entire candidate rule base.

Following the above, $\mathrm{R}_1^*, \mathrm{R}_2^*, \ldots, \mathrm{R}_{N_k^*}^*$ are used to compute the system's overall output $\hat{y}_k$ in response to the present input, resulting in:

$$\hat{y}_k = f(x_k) = \sum_{n=1}^{N_k^*}\lambda_{n,k}^* y_{n,k}^* = \sum_{n=1}^{N_k^*}\lambda_{n,k}^* \bar{x}_k^T a_{n,k-1}^*; \tag{14}$$

where $y_{n,k}^*$ is the output of $\mathrm{R}_n^*, n = 1, \ldots, N_k^*$, with the respective normalised firing strength calculated by:

$$\lambda_{n,k}^* = \frac{\mu_n^*(x_k)}{\sum_{j=1}^{N_k^*}\mu_j^*(x_k)}. \tag{15}$$

Stage 3. System Structure Updating.

In this stage, a check is first performed to see whether the new input x_k represents an unfamiliar pattern that is distinctive from any of the previously learned ones. This forms **Condition 1** below:

$$\begin{array}{l} \text{Cond.1 : If } (\mu_1^*(x_k) < \mu_0) \\ \text{Then } (x_k \text{ initiates a new fuzzy rule}) \end{array} \tag{16}$$

where $\mu_0 = e^{-1}$, which corresponds to the so-called "one sigma" rule [11]. The rationale behind this check is that if **Condition 1** is satisfied, x_k is out of the area influenced by the nearest prototype, showing a significant departure from the currently best-fitting local model in the system. Thus, the SAFL system needs to initialise a new fuzzy rule to incorporate the new pattern introduced by x_k. Note that μ_0 acts as a threshold in **Condition 1** in an effort to determine whether x_k is distant from the nearest prototype or not, and is not a adjustable parameter.

If **Condition 1** is met, a new cluster, $\mathbf{C}_N$ $(N \leftarrow N+1)$ is added to the system with x_k as its prototype according to Eqn. (9). A new fuzzy rule, R_N is initialised in the same form as Eqn. (1) with p_N as its premise parameters (noting the earlier operation of $N \leftarrow N+1$). The firing strength that R_N incurs with regard to x_k is set as $\mu_N(x_k) = 1$. The consequent parameters of R_N are therefore, initialised as below, following the work of [3,20]:

$$a_{N,k} \leftarrow \frac{1}{N-1}\sum_{i=1}^{N-1} a_{i,k-1}; \boldsymbol{\Theta}_{N,k} \leftarrow \Omega_0 \mathrm{I}_{(L+1)\times(L+1)}. \tag{17}$$

Else, x_k is assigned to the cluster, $\mathbf{C}_1^*$, namely, $\mathbf{C}_1^* \leftarrow \mathbf{C}_1^* \cup \{x_k\}$, and the meta–parameters of $\mathbf{C}_1^*$ are updated with:

$$\begin{array}{l} S_{1,k}^* = S_{1,k-1}^* + 1; c_{1,k}^* = c_{1,k-1}^* + \frac{x_k - c_{1,k-1}^*}{S_{1,k}^*}; \\ X_{1,k}^* = X_{1,k-1}^* + \frac{||x_k||^2 - X_{1,k-1}^*}{S_{1,k}^*}. \end{array} \tag{18}$$

The firing strength, $\mu_1^*(x_k)$ given by R_1^* is then re-calculated using Eqn. (12). The meta–parameters of other fuzzy rules and the associated clusters remain the same for the next learning cycle.

Stage 4. Fuzzy Rule Quality Monitoring. In this stage, stale fuzzy rules that contribute little to the system outputs computed so far are identified and removed from the candidate rule base, thereby implementing a procedure of rule pruning. The frequently used metrics for identifying stale fuzzy rules include *utility* [3], *age* [28], *rule importance* [29], *rule influence* [24], etc. SAFL uses a novel metric called average firing strength, which is closely related to the concept of *utility*. *Utility* is calculated as an average of normalised firing strengths produced by the individual fuzzy rules. Due to the normalisation operation (see Eqn. (4)), *utility* may hide or lose important information about the spatial similarities between input samples and local models represented by the individual fuzzy rules. The proposed metric is computed as an average of the firing strengths, being a direct measure of fitness of input samples to the local models and hence, it is more objective than *utility*.

To implement this, the average firing strength of each individual fuzzy rule (except the one newly established at the current learning cycle, if there is any) is first updated by:

$$M_{n,k} = M_{n,k-1} + \frac{\mu_{n,k}(x_k) - M_{n,k-1}}{I_n - k + 1}. \tag{19}$$

where $n = 1, \ldots, N$.

After this, **Condition 2** is then checked to identify the stale rules:

$$\begin{array}{ll} \text{Cond.2 :} & \text{If}(M_{n,k} < M_0) \\ & \text{Then}(\mathrm{R}_n \text{ and } \mathbf{C}_n \text{ are removed}) \end{array} \tag{20}$$

where M_0 is an externally controlled parameter determining the tolerance of SAFL towards stale fuzzy rules. If a particular fuzzy rule satisfies **Condition 2**, it implies that the local model represented by this rule does not fit the current input samples anymore and thus, needs to be removed. In this work, $M_0 = 0.05$ to assume no prior domain-dependent knowledge.

Before entering the final stage of the current learning cycle, the firing strengths, $\mu_n(x_k), n = 1, \ldots, N$ are re-ranked in descending order as the removal operation may lead to changes in system structure. Following this, the currently activated fuzzy rules, $\mathrm{R}_1^*, \mathrm{R}_2^*, \ldots, \mathrm{R}_{N_k^*}^*$ are re-selected from the rule base accordingly.

Stage 5. Local Consequent Parameter Updating SAFL employs a novel consequent parameter updating mechanism that only updates the consequent parameters of a dynamically changing set of activated fuzzy rules fitting the current input sample. This novel mechanism avoids unnecessary computations on updating those fuzzy rules with a marginal rule-firing strength, significantly improving the overall efficiency. Meanwhile, it aligns closely to the spirit of approximate modelling via achieving a soft and smooth adaptation of selected model parameters at each learning cycle.

The consequent parameters of the fuzzy rules survived from **Stage 4** are updated individually by the fuzzily weighted RLS algorithm [1] as given by Eqns. (21) and (22):

$$\boldsymbol{\Theta}_{n,k}^* = \boldsymbol{\Theta}_{n,k-1}^* - \frac{\lambda_{n,k}^* \boldsymbol{\Theta}_{n,k-1}^* \bar{x}_k \bar{x}_k^T \boldsymbol{\Theta}_{n,k-1}^*}{1 + \lambda_{n,k}^* \bar{x}_k^T \boldsymbol{\Theta}_{n,k-1}^* \bar{x}_k}; \tag{21}$$

$$a^*_{n,k} = a^*_{n,k-1} - \lambda^*_{n,k}\mathbf{\Theta}^*_{n,k}\bar{x}_k(\bar{x}_k^T a^*_{n,k-1} - y_k); \tag{22}$$

where $n = 1, \ldots, N^*_k$; and $a^*_{n,k}$ and $\mathbf{\Theta}^*_{n,k}$ are the consequent parameters and covariance matrix of R^*_n, respectively. The consequent parameters of those inactivated fuzzy rules at the current learning cycle are unchanged. From here, the SAFL learning process iterates, going back to **Stage 2** and starting the next cycle.

Remark 1. The rule selection mechanism used by SAFL works in a similar manner as the dropout technique used by deep neural networks [44] in the sense that both approaches drop out certain units (rules/nodes) during training, thereby improving the computational efficiency and reducing overfitting. The key difference is that the proposed rule selection mechanism drops out only those fuzzy rules that are less activated by the current input sample, and the number of fuzzy rules being dropped out changes with respect to different input samples, while the dropout technique randomly selects a fixed percentage of nodes to drop out.

Remark 2. Both the rule selection and rule pruning mechanisms effectively help to improve the overall computational efficiency of SAFL. Rule pruning works at the system level by removing the stale fuzzy rules from the rule base, enabling the system to maintain a set of quality fuzzy rules. Once a fuzzy rule is removed, it will no longer contribute to producing the system output nor be updated. In contrast, rule selection works at the sample level. During each learning cycle, it selects a dynamically changing set of activated fuzzy rules for system output generation and parameter updating. Whether a fuzzy rule is activated or not during the learning cycle is determined by the similarity between the input sample and its prototype. Such a rule selection mechanism brings forward two benefits: *a)* it significantly reduces the computational complexity of consequent parameter updating because a smaller number of fuzzy rules will be subsequently updated; and *b)* it improves the system's prediction precision because those less activated fuzzy rules would otherwise bring no improvement to the prediction precision but lead to overfitting.

In summary, the adaptive construction procedure for learning SAFL is presented in Algorithm 1.

Algorithm 1: SAFL construction

```
while (new input, x_k is available) do
  if (k = 1) then
    N ← 1
    initialise υ and χ by (8);
    initialise C_N and R_N by (9) and (10);
  else
    update υ and χ by (11);
    calculate μ_n(x_k) (n = 1,2,...,N) by (12);
    calculate ŷ_k by (14);
    if (x_k satisfies Condition 1)
      N ← N + 1;
      initiate C_N by (9);
      initiate R_N by (17);
      set μ_N(x_k) = 1;
    else
      update C*_1 by (18);
      update μ*_1(x_k) by (12);
    end if
    update M_{n,k} (n = 1,2,...,N) by (19);
    for n = 1 to N do
      if (R_n satisfies Condition 2) then
        remove R_n and C_n;
        N ← N − 1;
        end if
      end for
    for n = 1 to N*_k do
      update a*_{n,k} and Θ*_{n,k} by (21) and (22);
    end for
  end if
end while
```

3.3. Computational complexity

The computational complexity of the proposed approach is analysed here. As the SAFL system structure is dynamically self-evolving in response to incoming streaming data, in general, computational complexity varies at every learning cycle. Thus, the present analysis is conducted at the k^{th} time instance when x_k is observed, to attain generality.

Stage 1 considers system initialisation only and is not performed for any $k > 1$. Thus, the computational complexity of **Stage 1** is negligible within the overall learning process. **Stage 2** produces the system output. The complexity for updating the global meta–parameters, υ and χ, is $O(L)$. The complexity for calculating the firing strengths of all the present rules is $O(LN)$, for the respective firing strengths of those activated fuzzy rules is $O(N_k^*)$, and for the overall fuzzy output is $O(LN_k^*)$. **Stage 3** is for system structure updating, and the complexity for adding or updating a fuzzy rule is $O(L)$. The stale fuzzy rules are identified and removed at **Stage 4**, whose complexity is determined by that for updating the average firing strengths of all the current fuzzy rules, that is $O(N)$. In **Stage 5**, only the consequent parameters of the activated fuzzy rules are updated. Hence, the computational complexity is $O((L+1)^2 N_k^*)$.

Together, the computational complexity of SAFL at each learning cycle is therefore, $O((L+1)^2 N_k^*)$, and that of the overall learning process (given k observed input samples) is $O((L+1)^2 \sum_{j=1}^{k} N_j^*)$.

4. Stability analysis

The following two theorems guarantee the stability of the proposed SAFL fuzzy system.

Theorem 1. *If the error, denoted as e_k, between the system output and the desired output converges to a small neighbourhood of zero, and the average prediction error of the SAFL system is upper bounded for an infinite amount of samples ($k \to \infty$) as given by Eqn. (23), the stability of the SAFL system is ensured.*

$$\lim_{k \to \infty} \frac{1}{k} \sum_{j=1}^{k} e_j^2 \leqslant E_0; \tag{23}$$

where E_0 is a constant linearly correlated to the inherent approximation error of SAFL.

Proof. According to the universal approximation property of fuzzy systems [47], there exits a set of optimal consequent parameters, denoted by $\{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ that approximate the nonlinear function: $y_k = f(x_k)$. For generality, the optimal input–output relation of the SAFL system as given by Eqn. (7) is reformulated in Eqn. (24):

$$y_k = f(x_k) = \sum_{n=1}^{N} (\lambda_{n,k} \bar{x}_k^T \alpha_n + \varepsilon_n); \tag{24}$$

where ε_n is the inherent approximation error of R_n, and there is $\lambda_{n,k} \geqslant 0, n = 1, \ldots, N$.

Remark 3. Eqn. (24) is a general expression of the input–output relation of the SAFL system equivalent to Eqn. (7), where only N^* rules are fired to compute the system's output though the rule base contains N rules.

This is because in SAFL, if any rule R_n is not activated by x_k, it will make zero contribution to the current system output as $\lambda_{n,k} = 0$. Note that for conventional EFSs, $\lambda_{n,k}$ is guaranteed to be greater than 0. As such, SAFL can be considered as a special case of conventional EFSs (since now, $\lambda_{n,k} \geqslant 0$). Thus, this proof is valid for both conventional EFSs and SAFL without explicitly referring to the use of only N^* rules hereafter.

Based on Eqn. (24), the overall inherent approximation error of the system is $\epsilon_0 = \sum_{n=1}^{N} \lambda_{n,k} \varepsilon_n$. Since the SAFL system structure is dynamically self-evolving, the overall inherent approximation error can be reduced arbitrarily by increasing the number of fuzzy rules within the rule base, it is reasonable to assume that ϵ_0 is bounded with a constant τ_0 [8,34,40]:

$$|\epsilon_0| = \left| \sum_{n=1}^{N} \lambda_{n,k} \varepsilon_n \right| \leqslant \sum_{n=1}^{N} |\lambda_{n,k} \varepsilon_n| \leqslant \tau_0. \tag{25}$$

Considering the two following constraints:

$$\begin{cases} \sum_{n=1}^{N} \lambda_{n,k} = 1; \\ \lambda_{n,k} \geqslant 0, \forall n \in \{1, \ldots, N\}; \end{cases} \tag{26}$$

it can be concluded that the inherent approximation error of each fuzzy rule R_n is also bounded by the same constant $\tau_0, n = 1, \ldots, N$:

$$|\varepsilon_n| \leqslant \tau_0. \tag{27}$$

Combining Eqns. (14) and (24), the prediction error of the system is formulated as:

$$e_k = y_k - \hat{y}_k = \sum_{n=1}^{N} \lambda_{n,k} e_{n,k} = \sum_{n=1}^{N} \lambda_{n,k}(\bar{x}_k^T \tilde{a}_{n,k-1} + \varepsilon_n); \tag{28}$$

where $\tilde{a}_{n,k-1} = \alpha_n - a_{n,k-1}; e_{k,n}$ is the prediction error of R_n given input x_k, and there is $e_{n,k} = \bar{x}_k^T \tilde{a}_{n,k-1} + \varepsilon_n$. By setting $\mathrm{W}^{-1} = \mathbf{\Theta}_{n,k-1}, \mathrm{Y} = \sqrt{\lambda_{n,k}}\bar{x}_k$ and $\mathrm{Z}^{-1} = 1$ [34], Eqn. (29) can be derived from Eqn. (21):

$$\mathbf{\Theta}_{n,k}^{-1} = \mathbf{\Theta}_{n,k-1}^{-1} + \lambda_{n,k}\bar{x}_k\bar{x}_k^T. \tag{29}$$

Furthermore, the following can be derived from Eqns. (21) and (22):

$$\mathbf{\Theta}_{n,k}\bar{x}_k = \beta_{n,k}\mathbf{\Theta}_{n,k-1}\bar{x}_k; \tag{30}$$

$$\tilde{a}_{n,k} = \tilde{a}_{n,k-1} - \beta_{n,k}\lambda_{n,k}\mathbf{\Theta}_{n,k-1}\bar{x}_k e_{n,k}. \tag{31}$$

where $\beta_{n,k} = \frac{1}{1+\lambda_{n,k}\bar{x}_k^T\mathbf{\Theta}_{n,k-1}\bar{x}_k}$.

By defining $T_{n,k}$ as per the work of [34], such that

$$T_{n,k} = \tilde{a}_{n,k}^T \mathbf{\Theta}_{n,k}^{-1} \tilde{a}_{n,k}, \tag{32}$$

and combining it with Eqns. (30) and (31), it follows that

$$\begin{aligned} T_{n,k} &= \tilde{a}_{n,k}^T(\mathbf{\Theta}_{n,k-1}^{-1} + \lambda_{n,k}\bar{x}_k\bar{x}_k^T)\tilde{a}_{n,k} \\ &= T_{n,k-1} - 2\beta_{n,k}\lambda_{n,k}\bar{x}_k^T\tilde{a}_{n,k-1}e_{n,k} + \beta_{n,k}^2\lambda_{n,k}^2\bar{x}_k^T\mathbf{\Theta}_{n,k-1}\bar{x}_k e_{n,k}^2 + \lambda_{n,k}(\bar{x}_k^T\tilde{a}_{n,k})^2; \end{aligned} \tag{33}$$

where $T_{n,k-1} = \tilde{a}_{n,k-1}^T \mathbf{\Theta}_{n,k-1}^{-1} \tilde{a}_{n,k-1}$.

Although Eqn. (33) appears to be complex, it can be significantly simplified with the assistance of Eqns. (34)–(36), which are derived from Eqns. (28) and (31).

$$\bar{x}_k^T\tilde{a}_{n,k-1} = e_{n,k} - \varepsilon_n; \tag{34}$$

$$\lambda_{n,k}\bar{x}_k^T\mathbf{\Theta}_{n,k-1}\bar{x}_k = \frac{1}{\beta_{n,k}} - 1; \tag{35}$$

$$\begin{aligned} \bar{x}_k^T\tilde{a}_{n,k} &= \bar{x}_k^T\tilde{a}_{n,k} - \beta_{n,k}\lambda_{n,k}\bar{x}_k^T\mathbf{\Theta}_{n,k-1}\bar{x}_k e_{n,k} \\ &= \beta_{n,k}e_{n,k} - \varepsilon_n. \end{aligned} \tag{36}$$

Therefore, by substituting Eqns. (34)–(36) into Eqn. (33) and considering $\beta_{n,k} > 0$ (because $\bar{x}_k^T\mathbf{\Theta}_{n,k-1}\bar{x}_k > 0$), the following inequality can be derived from Eqns. (27) and (33):

$$\begin{aligned} T_{n,k} &= T_{n,k-1} - \lambda_{n,k}\beta_{n,k}e_{n,k}^2 + \lambda_{n,k}\varepsilon_n^2 \\ &\leqslant T_{n,k-1} - \lambda_{n,k}\beta_{n,k}e_{n,k}^2 + \lambda_{n,k}\tau_0^2. \end{aligned} \tag{37}$$

From this, the following inequality can be further derived:

$$\lambda_{n,k}\beta_{n,k}e_{n,k}^2 - \lambda_{n,k}\tau_0^2 \leqslant T_{n,k-1} - T_{n,k}. \tag{38}$$

After summing up both sides of the above inequality from the very first time instance to the current k^{th} time instance, the following inequality results [34]:

$$\sum_{j=1}^{k}(\lambda_{n,j}\beta_{n,j}e_{n,j}^2 - \lambda_{n,j}\tau_0^2) \leqslant T_{n,0} - T_{n,k}. \tag{39}$$

When there is $k \to \infty$, this leads to the inequality below (since $\lim_{k\to\infty}\frac{T_{n,0}-T_{n,k}}{k} = 0$):

$$\lim_{k\to\infty}\frac{1}{k}\sum_{j=1}^{k}\lambda_{n,j}\beta_{n,j}e_{n,j}^2 \leqslant \lim_{k\to\infty}\frac{1}{k}\sum_{j=1}^{k}\lambda_{n,j}\tau_0^2. \tag{40}$$

Considering Eqn. (28) with the constraints stated in Eqn. (26), the following relationship can be further derived, with $j = 1, \ldots, k$:

$$\begin{aligned}&\sum_{n=1}^{N}\lambda_{n,j}\beta_{n,j}e_{n,j}^{2} \geqslant \beta_{N}^{*}\sum_{n=1}^{N}\lambda_{n,j}e_{n,j}^{2} \geqslant \beta_{N}^{*}\sum_{n=1}^{N}\lambda_{n,j}^{2}e_{n,j}^{2}\\&\geqslant \beta_{N}^{*}\Big(\sum_{n=1}^{N}\lambda_{n,j}e_{n,j}\Big)^{2} = \beta_{N}^{*}e_{j}^{2};\end{aligned} \tag{41}$$

where $\beta_N^* = \min_{\substack{j=1,2,\ldots,k\\ n=1,2,\ldots,N}} \beta_{n,j}$.

By combining inequalities (40) and (41), the following inequality is derived:

$$\begin{aligned}&\lim_{k\to\infty}\tfrac{1}{k}\sum_{j=1}^{k}e_{j}^{2} \leqslant \lim_{k\to\infty}\tfrac{1}{k\beta_N^*}\sum_{j=1}^{k}\sum_{n=1}^{N}\lambda_{n,j}\beta_{n,j}e_{n,j}^{2}\\&\leqslant \lim_{k\to\infty}\tfrac{1}{k\beta_N^*}\sum_{j=1}^{k}\sum_{n=1}^{N}\lambda_{n,j}\tau_0^2 = \tfrac{\tau_0^2}{\beta_N^*}.\end{aligned} \tag{42}$$

Remark 4. Based on inequality (42), it can be concluded that inequality (23) holds. The average prediction error of the SAFL system is therefore, upper bounded for an infinite amount of samples ($k \to \infty$) and its stability is guaranteed.

Remark 5. The assumption that the overall inherent approximation error ϵ_0 is bounded with the constant τ_0 is required to ensure the stability of SAFL, and this assumption is commonly made in the literature [8,21,34,40].

Theorem 2. *When a new fuzzy rule is added or an existing fuzzy rule is removed, the stability of SAFL is still guaranteed.*

Proof. Without losing generality, suppose that at a certain time instance, a new fuzzy rule, denoted by R_{N+1}, is added to the rule base in the same form as Eqn. (1). The average prediction error of the SAFL system for an infinite amount of samples ($k \to \infty$) is reformulated as follows:

$$\lim_{k\to\infty}\frac{1}{k}\sum_{j=1}^{k}e_{j}^{2} = \lim_{k\to\infty}\frac{1}{k}\sum_{j=1}^{k}\Big(\sum_{n=1}^{N+1}\lambda_{n,j}e_{n,j}\Big)^{2}. \tag{43}$$

By combining inequalities (40), (42) and (43), the following inequality can be derived:

$$\begin{aligned}&\lim_{k\to\infty}\tfrac{1}{k}\sum_{j=1}^{k}e_{j}^{2} \leqslant \lim_{k\to\infty}\tfrac{1}{k}\sum_{j=1}^{k}\sum_{n=1}^{N+1}\lambda_{n,j}e_{n,j}^{2}\\&\leqslant \lim_{k\to\infty}\tfrac{1}{k\beta_{N+1}^*}\sum_{j=1}^{k}\sum_{n=1}^{N+1}\lambda_{n,j}\beta_{n,j}e_{n,j}^{2} \leqslant \tfrac{\tau_0^2}{\beta_{N+1}^*}.\end{aligned} \tag{44}$$

where $\beta_{N+1}^* = \min_{\substack{j=1,2,\ldots,k\\ n=1,2,\ldots,N+1}} \beta_{n,j}$.

Alternatively, if an existing fuzzy rule is removed from the rule base at a particular time instance, and the number of fuzzy rules in the rule base is $N-1$ now, inequality (44) can be reformulated as:

$$\begin{aligned}&\lim_{k\to\infty}\tfrac{1}{k}\sum_{j=1}^{k}e_{j}^{2} \leqslant \lim_{k\to\infty}\tfrac{1}{k}\sum_{j=1}^{k}\sum_{n=1}^{N-1}\lambda_{n,j}e_{n,j}^{2}\\&\leqslant \lim_{k\to\infty}\tfrac{1}{k\beta_{N-1}^*}\sum_{j=1}^{k}\sum_{n=1}^{N-1}\lambda_{n,j}\beta_{n,j}e_{n,j}^{2} \leqslant \tfrac{\tau_0^2}{\beta_{N-1}^*}.\end{aligned} \tag{45}$$

where $\beta_{N-1}^* = \min_{\substack{j=1,2,\ldots,k\\ n=1,2,\ldots,N-1}} \beta_{n,j}$.

Remark 6. It can be concluded from inequalities (44) and (45) that adding or removing a fuzzy rule does not influence the stability of SAFL.

Remark 7. The presented stability proof is derived on the basis of the proposed SAFL system. However, it is also applicable to more general fuzzy models that are developed through structure learning and local consequent parameter adaptation via the application of fuzzily weighted RLS algorithm [1]. If interested, more details regarding the proof for global consequent parameter learning can be found in [34].

5. Experimental Investigation

5.1. Problem cases studied

Numerical experimental studies are herein reported in evaluation of the performance of the proposed SAFL system, based on a wide range of benchmark problems including: one Mackey–Glass time series prediction problem, two nonlinear system identification problems, three real-world regression problems, one QuantQuote second resolution market data prediction problem, one S&P 500 closing price prediction problem, and 15 real-world classification problems. These datasets are outlined below.

1) Mackey–Glass time series prediction: Mackey–Glass time series is generated from the following [36]:

$$\frac{dx_k}{dk} = \frac{0.2x_{k-\tau}}{1 + x_{k-\tau}^{10}} - 0.1x_k; \tag{46}$$

In this study, $\tau = 17$ and the initial state $x_0 = 1.2$. In the prediction process, the input vector, consisting of three past values and the current value $[x_{k-18}, x_{k-12}, x_{k-6}, x_k]^T$, is used by each of the compared learning algorithms to predict the value x_{k-85}. A total of 3000 training samples are extracted from the time period between $k = 201$ and $k = 3200$, and 500 testing samples are extracted from the time period between $k = 5001$ and $k = 5500$.

2) Nonlinear system identification – Case 1: This problem is described as [17,27]:

$$y_k = \frac{y_{k-1}y_{k-2}(y_{k-1} - 0.5)}{1 + y_{k-1}^2 + y_{k-2}^2} + u_{k-1}; \tag{47}$$

where $y_{-1} = y_0 = 0; u_{k-1} = \sin\left(\frac{2\pi(k-1)}{25}\right)$. In this study, $[y_{k-1}, y_{k-2}, u_{k-1}]^T$ and y_k are selected as the input and output of the nonlinear dynamic plant, respectively, to be learned by each of the learning algorithms compared. 5000 training samples are produced from the time period between $k = 1$ and $k = 5000$ and 200 testing samples are produced from the time period between $k = 5001$ and $k = 5200$.

3) Nonlinear system identification – Case 2: This problem is described as [6,38]:

$$x_k = \frac{19\beta_k}{40} \sin\left(\frac{16u_{k-1} + 8x_{k-1}}{\beta_k(3 + 4u_{k-1}^2 + 4x_{k-1}^2)}\right) + \frac{u_{k-1} + x_{k-1}}{5}; \tag{48}$$

where β_k is a time-varying parameter in accordance to the following equation:

$$\beta_k = \begin{cases} 1.0, & 0 \leqslant k \leq 150; \\ 0.9, & 1501 \leqslant k \leq 2500; \\ 0.8, & 2501 \leqslant k \leq 5000; \end{cases} \tag{49}$$

Training input u_k is uniformly taken from the range of $[-1, 1]$, while the testing input is given by $u_{k-1} = \sin\left(\frac{2\pi k}{25}\right)$. In particular, 5000 training samples and 200 testing samples are produced.

4) Real-world benchmark regression: For conciseness, key information of all three benchmark regression case studies [1] is summarised in Table 2.

5) QuantQuote second resolution market data prediction: This dataset contains tick-by-tick data on all NASDAQ, NYSE and AMEX securities from 1998 to the present [2], concerning the following attributes: i) time tag, k; ii) open price, $x_{k,1}$; iii) high price, $x_{k,2}$; iv) low price, $x_{k,3}$; and v) close price, $x_{k,4}$. There are 19,144 data samples involved in this problem (see Section 5.4 for ways of partitioning the dataset for training and testing). In this case study, four specific experiments are conducted as follows:

- Using the current four prices to predict the close price five steps ahead:

$$x_{k+5,4} = f([x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}]^T); \tag{50}$$

- Using the current four prices to predict the close price 10 steps ahead:

$$x_{k+10,4} = f([x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}]^T); \tag{51}$$

[1] Available at: https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html
[2] Available at: https://quantquote.com/historical-stock-data

Table 2
Regression Problems.

Dataset	#(Attributes)	#t(Training Samples)	#(Testing Samples)
Autos	15 inputs+1 output	80	79
California Housing	8 inputs+1 output	10320	10320
Delta Ailerons	5 inputs+1 output	3000	4129

- Using the current four prices to predict the close price 15 steps ahead:

$$x_{k+15,4} = f([x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}]^T); \tag{52}$$

- Using the current four prices to predict the close price 20 steps ahead:

$$x_{k+20,4} = f([x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}]^T). \tag{53}$$

As with the common practice in the literature [3,20], the four prices have been standardised online.

6) S&P500 closing price prediction: Daily closing prices of S&P500 are collected from the Yahoo! Finance website [3], ranging from 03/01/1950 to 12/03/2009, with 14893 samples in total. This dataset is further expanded with its flipped time series, and thus, 29786 samples after augmentation. As with common practice, all data has been normalised to the range of $[0, 1]$, with (normalised) samples within the first half of the augmented dataset, namely, the original time series, used for training and the remaining data, i.e., the flipped time series for online testing. The regression model is defined as follows:

$$x_{k+1} = f([x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1}, x_k]^T). \tag{54}$$

7) Real-world benchmark classification: For presentational simplicity, key information of the 15 classification problems adopted for this experimental study is outlined in Table 3. Particularly, the problems investigated include: nine classical benchmark ones[4], two real-world ones (Electricity Pricing[5] and Skin Segmentation[6]), two synthetic ones (Hyperplane and SEA[7]) and two visual ones (PMNIST and RMNIST[8]). Properties of the corresponding datasets are also described in the same table.

These datasets are widely used for performance evaluation in the literature, e.g., [3,5,7,13,20,38,49], thus, they are considered herein for evaluating the regression and classification performance of SAFL.

5.2. Experimental setup

In the experimental investigation, SAFL is compared against a variety of state-of-the-art algorithms in the literature. Algorithm performances on regression problems are evaluated by root mean square error (*RMSE*) and non-dimensional error index (*NDEI*):

$$RMSE = \sqrt{\frac{\sum_{j=1}^{k} e_j^2}{k}}; NDEI = \frac{RMSE}{\varrho_k}; \tag{55}$$

where $e_j = y_j - \hat{y}_j$ and ϱ_k is the standard deviation of $\{y\}_k = \{y_1, y_2, \ldots, y_k\}$.

Performances of the algorithms on classification problems are evaluated based on classification accuracy (*Acc*):

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}; \tag{56}$$

where TP, TN, FP and FN represent the resulting true positives, true negatives, false positives and false negatives, respectively.

Other performance indices utilised in the experimental studies are the number of fuzzy rules activated, denoted as $\#(Rules)$, and the consumption of training time, denoted as t_{exe} (in seconds).

All investigated algorithms are developed using MATLAB2019b, with experiments performed on a laptop with dual core i7 processor 2.60 GHz×2 and 16.0 GB RAM. The MATLAB code of SAFL is publicly available[9].

[3] Available at: https://uk.finance.yahoo.com/
[4] Available at: https://sci2s.ugr.es/keel/index.php
[5] Available at: https://moa.cms.waikato.ac.nz/datasets/
[6] Available at: https://archive.ics.uci.edu/ml/datasets/Skin + Segmentation
[7] Available at: https://scikit-multiflow.github.io/
[8] Available at: https://nlp.stanford.edu/projects/mer/
[9] Available at: https://github.com/Gu-X/Self-Adaptive-Fuzzy-Learning-System

Table 3
Classification Problems.

Dataset	#(Attributes)	#(Samples)	#(Classes)	Characteristics
Wine	13 inputs+1 label	178	3	stationary
Monk2	6 inputs+1 label	432	2	stationary
Diabetes	8 inputs+1 label	768	2	stationary
Balance	4 inputs+1 label	625	3	stationary
Vehicle	18 inputs+1 label	846	4	stationary
Pageblocks	10 inputs+1 label	5472	5	stationary
Texture	40 inputs+1 label	5500	11	stationary
Magic	10 inputs+1 label	19020	2	stationary
Penbased	16 inputs+1 label	10992	10	stationary
Electricity Pricing	8 inputs+1 label	45312	2	non-stationary
Skin Segmentation	3 inputs+1 label	245057	2	non-stationary
Hyperplane	4 inputs+1 label	120000	2	non-stationary
SEA	3 inputs+1 label	200000	2	non-stationary
PMNIST	784 inputs+1 label	65000	10	non-stationary
RMNIST	784 inputs+1 label	70000	10	non-stationary

5.3. Sensitivity analysis

The proposed SAFL system has three externally controlled parameters, namely, γ_0 (used in Eqn. (13)), Ω_0 (used in Eqns. (10) and (17)) and M_0 (used in **Condition 2**). The influence of the three externally controlled parameters on the system performance is therefore, first investigated here, using two real-world regression benchmark problems: *i)* Delta Ailerons; and *ii)* California Housing.

1) Influence of γ_0: In this experiment, the value of γ_0 is varied from 0.1 to 1.0 with a step of 0.1, and the other two parameters are set as: $\Omega_0 = 1000$ and $M_0 = 0.05$. The performance of SAFL, in terms of $RMSE, \#(Rules)$ and t_{exe}. are presented in Table 4. It can be observed from this table that γ_0 influences both the computational efficiency and prediction precision of SAFL. A greater value of γ_0 enables more fuzzy rules to be involved in system output generation and parameter updating, leading to higher computational complexity. Note that involving more fuzzy rules for system output generation does not necessarily result in better prediction performance. The best value range of γ_0 as suggested by this experiment is $[0.3, 0.7]$.

2) Influence of Ω_0: In this experiment, the value of Ω_0 is varied from 1 to 10000 with a nonlinear step size, and the other two parameters are set as: $\gamma_0 = 0.5$ and $M_0 = 0.05$. The performance of SAFL with different Ω_0 values are also presented in Table 4. The results show that, although Ω_0 has negligible influence upon computational efficiency, the prediction accuracy of SAFL deteriorates if the value of Ω_0 is set to be too small or too great. The recommended value range of Ω_0 is between 500 and 2000.

3) Influence of M_0: In this experiment, the value of M_0 is varied from 0 to 0.15, again with a nonlinear step size, and the other two parameters are set as: $\gamma_0 = 0.5$ and $\Omega_0 = 1000$. The experimental results are also tabulated in Table 4, showing that M_0 has a greater influence on the system complexity. A greater value of M_0 helps SAFL to remove stale fuzzy rules rapidly. However, this may bring a potential risk that the system can forget the learned knowledge from streaming data too fast, given the changing rate of removing the learned rules from the rule base. The recommended value range of M_0 is $[0.03, 0.10]$.

Based on the above initial investigation, in the following general experimental studies, the three externally controlled parameters are empirically set to: $\gamma_0 = 0.5, \Omega_0 = 1000$ and $M_0 = 0.05$.

5.4. Performance comparison

SAFL is herein compared with a number of state-of-the-art EFSs on a range of aforementioned datasets. Technical details of the compared EFSs are beyond the scope of this paper, but all can be easily found in the literature (see their references as given in Table 5).

1) On Mackey–Glass time series: For this widely-used chaotic time series problem, the following three performance measures are considered: $NDEI, \#(Rules)$, and t_{exe}, with the results given in Table 5, following the standard experimental protocol as per that is adopted in [6,20,38]. The results by alternative EFSs are obtained directly from [10,17,18,43] as well as those from [6,20,38] for fair comparison. The stepwise prediction error (e_k), the average prediction error over time ($E_k = \frac{1}{k}\sum_{j=1}^{k} e_j^2$) and the $\#(Rules)$ over the online learning process are shown in Figs. 2(a)-(c), respectively. It can be seen that SAFL is able to produce highly accurate prediction on this dataset with the lowest *NDEI* value (of 0.1048), surpassing its competitors, while its computational efficiency is the highest (taking only 0.2271 s to process 3000 training samples). Importantly, it can be observed from Fig. 2(c) that the average prediction error of SAFL gradually converges to zero overtime, which empirically verifies the correctness of the stability analysis given previously.

2) On benchmark nonlinear system identification cases: This experiment involves: *i*) nonlinear dynamic plant identification [17,20], and *ii*) time-varying nonlinear system identification [6,38], for performance comparison. Prediction results in terms

Table 4
Influence of Parameter Settings.

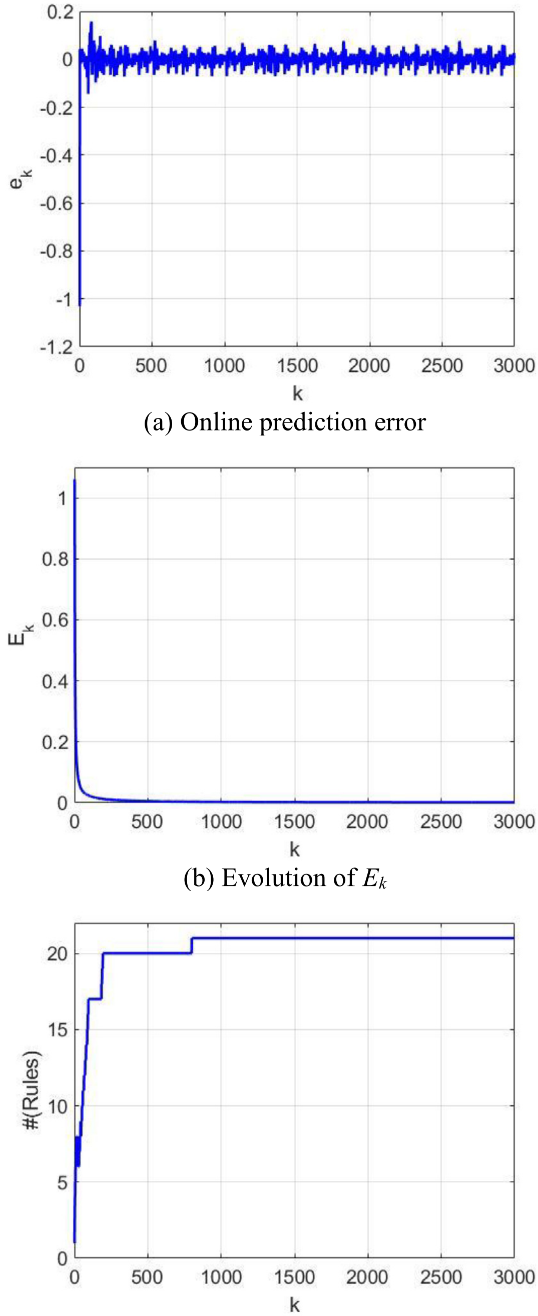
Dataset	Measure	γ_0 ($\Omega_0 = 1000; M_0 = 0.05$)								
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Delta Ailerons	*RMSE*	0.0501	0.0495	0.0492	0.0492	0.0492	0.0492	0.0493	0.0494	0.0497
	$\#(Rules)$	36	36	36	36	36	36	36	36	36
	t_{exe}	0.1356	0.1579	0.1994	0.2087	0.2620	0.2893	0.3488	0.4469	0.9910
California Housing	*RMSE*	0.0685	0.0682	0.0684	0.0692	0.0696	0.0703	0.0709	0.0717	0.0724
	$\#(Rules)$	26	26	26	26	26	26	26	26	26
	t_{exe}	0.4352	0.5086	0.6032	0.6316	0.7214	0.8411	1.0752	1.3298	2.7038
Dataset	**Measure**	Ω_0 ($\gamma_0 = 0.5; M_0 = 0.05$)								
		1	10	100	200	500	1000	2000	5000	10000
Delta Ailerons	*RMSE*	0.0603	0.0513	0.0491	0.0492	0.0492	0.0492	0.0492	0.0493	0.0493
	$\#(Rules)$	36	36	36	36	36	36	36	36	36
	t_{exe}	0.2107	0.2090	0.2084	0.2089	0.2089	0.2087	0.2086	0.2089	0.2079
California Housing	*RMSE*	0.0866	0.0765	0.07	0.0695	0.0693	0.0692	0.0692	0.0691	0.0691
	$\#(Rules)$	26	26	26	26	26	26	26	26	26
	t_{exe}	0.6426	0.6539	0.6258	0.6262	0.6271	0.6316	0.6323	0.6314	0.6810
Dataset	**Measure**	M_0 ($\gamma_0 = 0.5; \Omega_0 = 1000$)								
		0	0.01	0.02	0.03	0.04	0.05	0.07	0.10	0.15
Delta Ailerons	*RMSE*	0.0486	0.0487	0.0488	0.0493	0.0493	0.0492	0.0495	0.0535	0.057
	$\#(Rules)$	99	75	62	47	44	36	29	18	10
	t_{exe}	0.3038	0.3021	0.2538	0.2327	0.2227	0.2087	0.1950	0.1649	0.1406
California Housing	*RMSE*	0.0676	0.0685	0.0683	0.0682	0.0691	0.0692	0.0696	0.0712	0.0799
	$\#(Rules)$	51	42	36	35	30	26	23	19	11
	t_{exe}	0.6980	0.6696	0.6480	0.6558	0.6499	0.6316	0.5838	0.5901	0.5055

Table 5
Performance Comparison on Mackey–Glass Time Series Problem.

Algorithm	*NDEI*	$\#(Rules)$	t_{exe}
SAFL	**0.1048**	20	**0.1868**
eTS [1]	0.2141	**4**	21.9745
SAFIS [36]	0.2925	**4**	0.4375
OS-Fuzzy-ELM [35]	0.2991	5	0.9253
ESAFIS [37]	0.2487	6	2.7656
PANFIS [29]	0.2847	33	4.8679
GENEFIS [30]	0.1198	42	4.9694
eFuMo [9]	0.1388	41	-
ALMMo [3]	0.4020	9	0.4688
CENFS [6]	0.2635	5	0.4368
LEOA [17]	0.2480	42	144.7818
SEFS [18]	0.1287	**4**	0.3510
PALM [14]	0.1380	18	0.7771
RMCEFS [38]	0.1172	5	0.3432
eGAUSS+ [43]	0.2728	25	~5
EFS-SLAT [19]	0.1140	8	-
HiPCA [10]	0.2495	25	~14
PSO-ALMMo* [20]	0.1910	8	314.3214

of *RMSE* and $\#(Rules)$ are reported in Table 6, again, by following the standard experimental protocol in the relevant literature. The reported results by eTS, Simpl_eTS, CENFS and RMCEFS are obtained from [6,35,38]. It can be seen that SAFL produces the most accurate prediction with the least *RMSE* (0.0052 for the first problem and 0.0955 for the second), outperforming the rest. The system complexity of SAFL measured by $\#(Rules)$ is also relatively lower than those of its counterparts. Furthermore, the training times of SAFL on the two nonlinear problems are 0.2329 s and 0.1878 s, respectively, showing that SAFL is of very high computational efficiency.

3) On real-world regression problems: Three benchmark regression cases, as listed in Table 2, are considered in this experiment, and performances of compared algorithms are again measured by *RMSE*, $\#(Rules)$ and t_{exe}, as reported in Table 7, following the standard experimental protocol [6,20,38]. It can be seen that SAFL outperforms alternative EFSs on the Delta Ailerons and California Housing datasets in terms of *RMSE*, and that its prediction accuracy ranks third on the Auto dataset. Interestingly, whilst SAFL identifies more fuzzy rules during the training process, its computational efficiency is the highest



(a) Online prediction error
(b) Evolution of E_k
(c) Evolution of #*(Rules)*

Fig. 2. Prediction result on Mackey–Glass time series problem.

among all the algorithms. This is owing to the fact that in each learning cycle, only the activated fuzzy rules are involved for system output generation and parameter updating. The online prediction errors of SAFL on the three benchmark problems are plotted in Fig. 3, the average prediction errors overtime in Fig. 4, and the numbers of fuzzy rules over time in Fig. 5. It is clear from Fig. 4 that the average prediction error of SAFL gradually converges to a very small value close to zero, given more training samples. This further empirically proves the stability of SAFL.

The significance of performance improvement enabled by SAFL over five selected EFS approaches is analysed by running statistical pairwise t-tests. Table 8 presents the test outcomes, in terms of p-value. It can be observed that the majority of the

Table 6
Performance on Nonlinear System Identification.

Algorithm	Nonlinear System 1		Nonlinear System 2	
	RMSE	#(*Rules*)	*RMSE*	#(*Rules*)
SAFL	**0.0052**	12	**0.0955**	9
eTS	0.0350	**3**	0.1360	**6**
Simpl_eTS	0.0225	22	0.2506	34
SAFIS	0.0408	10	0.2423	11
ESAFIS	0.0126	5	0.1783	14
ALMMo	0.0441	9	0.2354	11
CENFS	0.0183	12	0.1736	19
RMCEFS	-	-	0.1172	10
PSO-ALMMo*	0.0148	7	0.1204	11

Table 7
Performance on Real-World Regression Problems.

Algorithm	Auto			Delta Ailerons			California Housing		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	0.0449	22	**0.0045**	**0.0492**	36	**0.2087**	**0.0692**	28	**0.6316**
DENFIS	0.4516	8	-	0.0497	11	-	0.0715	14	-
eTS	0.0535	3	0.2184	0.0513	4	2.1372	0.0772	3	7.6128
Simpl_eTS	0.0689	10	0.5772	0.0512	4	2.0088	0.0773	3	5.5536
SAFIS	0.1184	5	0.4524	0.0549	14	6.8328	0.0988	12	21.9805
OS-Fuzzy-ELM	0.0595	**2**	0.0296	0.0507	3	0.4602	0.1320	5	6.7252
ESAFIS	0.0604	3	0.2184	0.0506	13	12.3865	0.0892	6	30.2330
McFIS	0.0687	3	-	0.0509	15	-	0.0822	15	-
ALMMo	0.0565	8	0.0591	0.0513	10	0.3651	0.0782	10	1.2423
CENFS	0.0666	**2**	0.0156	0.0502	3	0.3432	0.0878	**2**	1.5444
RMCEFS	**0.0409**	**2**	0.0156	0.0498	**2**	0.2184	-	-	-
PSO-ALMMo*	0.0425	8	12.6411	0.0497	10	362.3939	0.0711	11	1340.5226

p values returned by the tests are below the level of significance specified by $\alpha = 0.05$, suggesting that the performance of SAFL is significantly better than the majority of the alternatives.

4) On QuantQuote second resolution market data prediction: All four prediction models as given for this dataset in Section 5.1 are considered here. During the experimental study, this dataset is divided into ten subsets evenly following the timeline. Each time, nine of the ten subsets are randomly selected for training and the remaining one is used for testing. Statistical results obtained by SAFL, eTS, SAFIS, ESAFIS, ALMMo, PALM and PSO-ALMMo after 25 Monte Carlo simulations are tabulated in Table 9. It can be observed from the results that SAFL surpasses all conventional EFSs on this problem, in both prediction accuracy and computational efficiency.

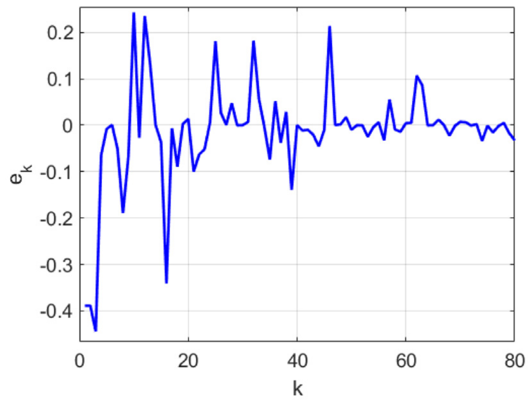
5) On S&P 500 closing price prediction: For this problem, the prediction results of SAFL in terms of *NDEI* and $\#(Rules)$ are tabulated in Table 10 following the commonly used experimental protocal [18], together with the results of alternative EFSs for comparison. These results show that SAFL outperforms all the others in prediction accuracy: It has the lowest *NDEI* value (of 0.0121) among all. Additionally, the entire learning process of the SAFL system only costs 1.8101 s, which means that it merely requires 61 microseconds on average to process each data sample, given the 29,786 samples in total. For better interpretability, ten of the fuzzy rules generated by SAFL during the learning process are listed in Table 11 as examples.

6) On classification: Performance of SAFL is firstly tested on the nine benchmark datasets and compared with the following three state-of-the-art fuzzy classifiers including D-MOFARC [12], HID-TSK-FC [49] and CFBLS [13]. For binary datasets, the class labels of the data samples are set to be either -1 or 1. For multi-class datasets, the one-against-all strategy is used [49]. Since SAFL is a MISO system, the predicted class label of a given input sample is determined by the following:

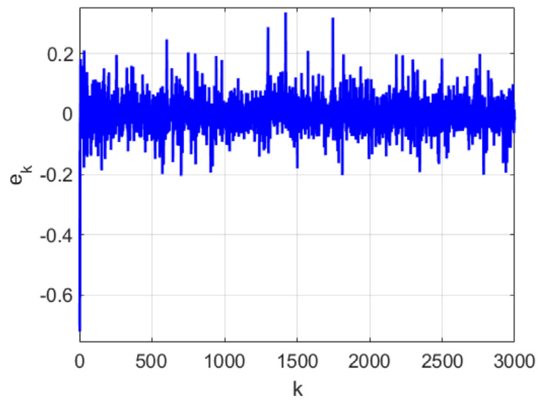
$$\hat{y}_k = \begin{cases} -1, & f(x_k) < 0. \\ 1, & f(x_k) \geqslant 0. \end{cases} \quad (57)$$

Performance of SAFL obtained from ten-fold cross-validation is summarised in Table 12, in terms of *Acc* and $\#(Rules)$. Performances of the three aforementioned comparative models directly obtained from [12,13,49] are reported in the same table for comparison. Table 12 shows the superiority of SAFL over the three competitors in terms of classification precision, outperforming them on six out of nine benchmark problems.

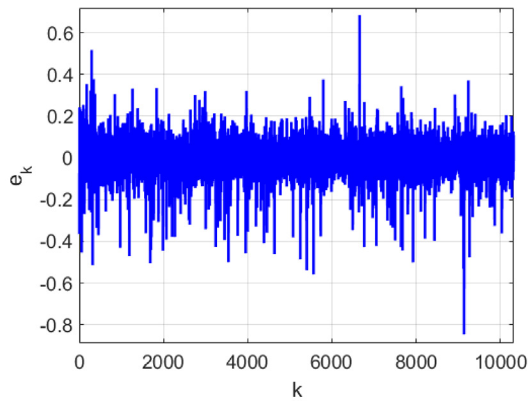
The two real-world classification problems are then used for evaluating the capability of SAFL on handling real-world uncertainties. In both case studies, 80% of the samples are randomly selected for training and the remaining 20% for testing. Performances of SAFL and seven comparative EFSs are summarised in Table 13, in terms of $Acc, \#(Rules)$ and t_{exe}. These



(a) Auto



(b) Delta Ailerons



(c) California Housing

Fig. 3. Online prediction error on real-world regression problems.

results are obtained after 10 Monte Carlo simulations to accommodate a certain degree of randomness. Table 13 shows that SAFL outperforms the others in terms of classification accuracy and computational efficiency, on both problems. To examine the statistical significance of the performance improvement of SAFL over the rest, Wilcoxon signed rank tests are carried out. Table 14 presents the test outcomes, in terms of p-value and z-score, for which the classification results by each individual algorithm across 10 Monte Carlo simulations are used. It can be observed that the p-values returned by the tests are all approximately zero, far below the level of significance specified by $\alpha = 0.05$. Thus, the null hypothesis is rejected, suggesting that the performance of SAFL is significantly better than the other algorithms.

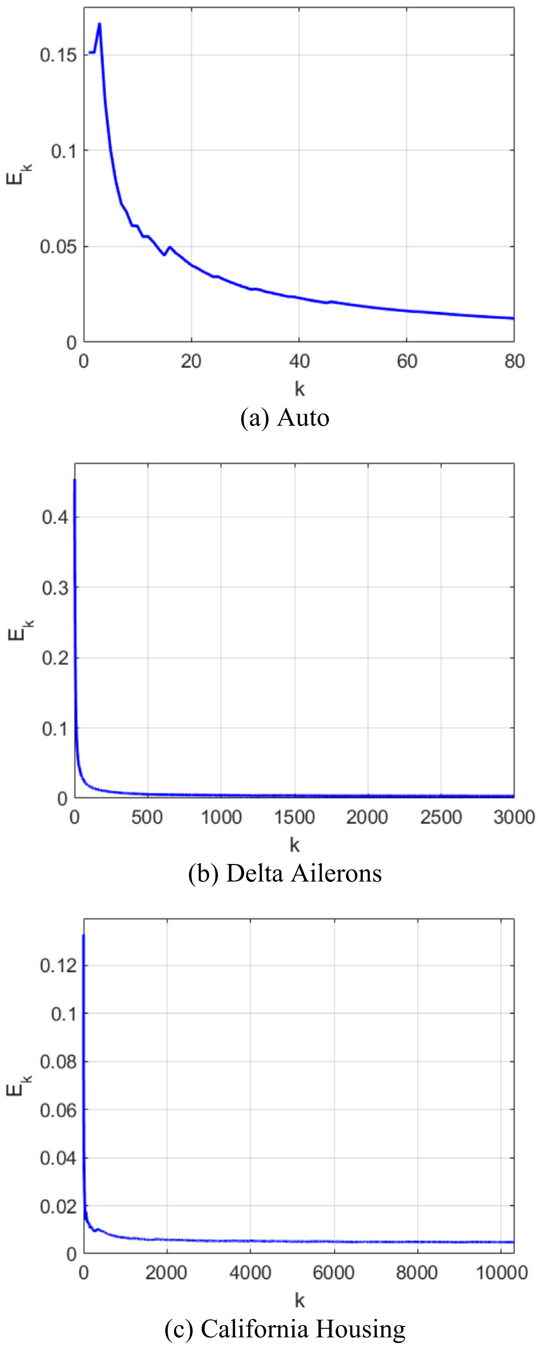


Fig. 4. Evolution of E_k on real-world regression problems.

5.5. Ablation analysis

The ablation study is conducted herein to demonstrate the advantages of the inference scheme and consequent parameter updating mechanism of the proposed SAFL system. In particular, three different fuzzy systems are implemented, denoted as SAFL-1, SAFL-2 and SAFL-3. SAFL-1 only employs the inference scheme of SAFL; that is, it uses the activated fuzzy rules for producing system outputs and updates the consequent parameters of all fuzzy rules within the rule base at each learning cycle. SAFL-2 keeps the same consequent parameter updating mechanism as SAFL but involves all fuzzy rules to

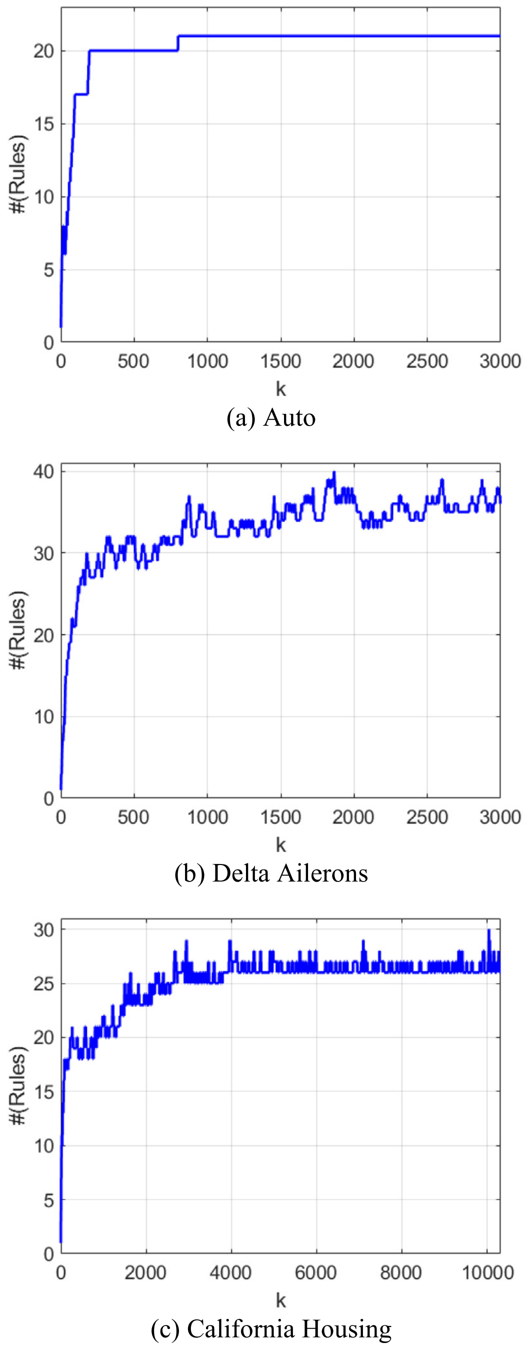


Fig. 5. Evolution of $\#(Rules)$ on real-world regression problems.

Table 8
p-Values in Statistical Pairwise *t*-Test.

SAFL vs	Autos	Delta Ailerons	California Housing
eTS	0.5409	0.9828	0.0286
SAFIS	0.2455	0.0000	0.0000
ESAFIS	0.2500	0.0001	0.0000
ALMMo	0.0184	0.0000	0.9415
PSO-ALMMo*	0.6729	0.1068	0.0424

Table 9
Comparison on Quantquote Second Resolution Market Dataset.

Algorithm	$x_{k+5,4} = f(x_k)$			$x_{k+10,4} = f(x_k)$		
	NDEI	#(*Rules*)	t_{exe}	*NDEI*	#(*Rules*)	t_{exe}
SAFL	**0.2785±0.1458**	18.4400±8.2162	**1.0096±0.2793**	**0.3342±0.1729**	17.1600±7.3410	**1.0194±0.3218**
eTS	0.2887±0.1435	28.8000±10.9316	144.3784±7.4217	0.3556±0.1955	23.0000±12.5532	138.3399±15.8391
SAFIS	0.9345±0.6703	21.2000±6.4743	2.9346±1.0486	0.8337±0.6703	22.2400±7.4234	3.0525±1.0827
ESAFIS	0.6376±0.7200	**2.2000±0.8165**	2.8869±0.5119	0.4338±0.2659	**2.3200±0.8524**	3.2894±0.8580
ALMMo	0.4065±0.2604	8.7600±2.9195	1.3578±0.0739	0.6131±0.5947	9.1200±2.3861	1.3691±0.0628
PALM	0.3346±0.1677	34.0000±25.6288	14.6401±10.6883	0.4565±0.3865	28.3200±30.5023	15.1544±18.7338
PSO-ALMMo*	0.2821±0.1253	8.0000±2.2174	1995.8353±290.6900	0.3351±0.1756	8.6400±2.3608	2151.0319±326.0238
Algorithm	$x_{k+15,4} = f(x_k)$			$x_{k+20,4} = f(x_k)$		
	NDEI	#(*Rules*)	t_{exe}	*NDEI*	#(*Rules*)	t_{exe}
SAFL	**0.4181±0.1516**	16.4400±6.7025	**0.8838±0.2381**	**0.5274±0.2033**	19.0400±6.7483	**1.0097±0.2495**
eTS	0.4187±0.1531	23.4400±12.2443	143.2560±12.8344	0.5610±0.2616	27.7200±12.1227	143.8742±8.6245
SAFIS	1.8092±1.3696	20.0800±8.9066	3.0836±1.6286	1.9117±1.7789	22.5200±8.7613	3.2456±1.4820
ESAFIS	1.0550±1.0052	**2.2800±1.1000**	3.3788±0.6885	1.3302±1.4781	**2.8400±1.1431**	3.2344±0.5149
ALMMo	0.9568±0.7819	9.0000±2.2174	1.3843±0.0919	1.2946±1.3748	8.3600±2.0388	1.3641±0.0855
PALM	0.9103±1.4751	37.2000±36.6772	19.9113±23.3570	0.6796±0.3083	52.8800±30.8954	25.4129±17.6463
PSO-ALMMo*	0.4255±0.1498	9.3400±2.7734	2232.5733±381.4395	0.5278±0.1963	8.6800±1.9088	2072.1742±260.7311

Table 10
Comparison on S&P 500 Closing Price Prediction.

Algorithm	*NDEI*	#(*Rules*)	t_{exe}
SAFL	**0.0121**	19	**1.8101**
PANFIS	0.09	4	55.3
GENEFIS	0.07	**2**	48.3
eT2RFNN [31]	0.04	**2**	7.26
ALMMo	0.0149	7	2.5766
LEOA	0.1229	52	-
SEFS	0.0182	**2**	2.3467
EFS-SLAT	0.0156	23	-
PSO-ALMMo*	0.0146	6	2783.3210

Table 11
Fuzzy Rules Identified by SAFL for S&P500 Closing Price Prediction Problem.

#	**Fuzzy Rule**
R_1	IF$(x \sim [0.0175, 0.0176, 0.0180, 0.0182, 0.0180]^T)$THEN$(y_n = 0.0017 + 0.1518x_1 + 0.1624x_2 + 0.1754x_3 + 0.1947x_4 + 0.2242x_5)$
R_2	IF$(x \sim [0.0227, 0.0229, 0.0228, 0.0230, 0.0233]^T)$THEN$(y_n = 0.0034 + 0.1315x_1 + 0.1452x_2 + 0.1658x_3 + 0.1928x_4 + 0.2337x_5)$
R_3	IF$(x \sim [0.0290, 0.0293, 0.0291, 0.0293, 0.0293]^T)$THEN$(y_n = 0.0009 + 0.1188x_1 + 0.1401x_2 + 0.1770x_3 + 0.2292x_4 + 0.3099x_5)$
R_4	IF$(x \sim [0.0366, 0.0370, 0.0375, 0.0375, 0.0371]^T)$THEN$(y_n = 0.0008 + 0.0868x_1 + 0.1117x_2 + 0.1641x_3 + 0.2430x_4 + 0.3762x_5)$
R_5	IF$(x \sim [0.0462, 0.0463, 0.0464, 0.0463, 0.0465]^T)$THEN$(y_n = 0.0001 + 0.0346x_1 + 0.0713x_2 + 0.1400x_3 + 0.2558x_4 + 0.4967x_5)$
R_6	IF$(x \sim [0.0578, 0.0578, 0.0579, 0.0581, 0.0585]^T)$THEN$(y_n = 0.0001 + 0.0240x_1 + 0.0424x_2 + 0.0997x_3 + 0.2380x_4 + 0.5934x_5)$
R_7	IF$(x \sim [0.0703, 0.0713, 0.0725, 0.0729, 0.0729]^T)$THEN$(y_n = 0.0002 + 0.0256x_1 + 0.0419x_2 + 0.1144x_3 + 0.2423x_4 + 0.5730x_5)$
R_8	IF$(x \sim [0.0861, 0.0872, 0.0882, 0.0879, 0.0881]^T)$THEN$(y_n = 0.0027 + 0.0221x_1 + 0.0340x_2 + 0.1056x_3 + 0.2453x_4 + 0.5651x_5)$
R_9	IF$(x \sim [0.1418, 0.1416, 0.1420, 0.1449, 0.1454]^T)$THEN$(y_n = 0.0029 - 0.0004x_1 + 0.0058x_2 + 0.0650x_3 + 0.1888x_4 + 0.7225x_5)$
R_{10}	IF$(x \sim [0.1630, 0.1650, 0.1695, 0.1627, 0.1652]^T)$THEN$(y_n = -0.0000 + 0.0231x_1 - 0.0012x_2 + 0.0451x_3 + 0.1892x_4 + 0.7450x_5)$

generate the system outputs. SAFL-3 involves all fuzzy rules for system output generation and updates their consequent parameters at each learning cycle. Other than these particular specifications, the three fuzzy systems each utilise exactly the same computing mechanism as used by SAFL.

The performances of SAFL and its three specific derivatives are evaluated on five real-world benchmark problems (without incurring excessive computational costs that would otherwise be required to run these different systems against all datasets used previously). The results are summarised in Table 15. It can be observed from this table that the simplified inference and consequent parameter updating schemes used by the proposed SAFL system not only significantly improves its computational complexity ($2-4$ times faster than SAFL-3 across these benchmark problems), but also effectively reduces the prediction errors (e.g., leading to 51%, 88% and 35% less errors on Mackey–Glass and the two nonlinear system identification problems, respectively). Once again, this empirically illustrates the effectiveness and validity of the proposed approach.

Table 12
Comparison of SAFL, D-MOFARC, HID-TSK-FC and CFBLS on Nine Benchmark Datasets.

Dataset	SAFL		D-MOFARC		HID-TSK-FC		CFBLS	
	Acc	#(*Rules*)	*Acc*	#(*Rules*)	*Acc*	#(*Rules*)	*Acc*	#(*Rules*)
Wine	**0.9833±0.0268**	34.9	0.9580±0.0388	**8.6**	-	-	0.9740±0.0114	13.7
Monk2	**0.9541±0.0283**	36.5	-	-	0.6494±0.0120	7.0	0.9066±0.0149	**5.9**
Diabetes	0.7669±0.0375	38.9	0.7550±0.0447	10.4	0.7182±0.0156	9.0	**0.7764±0.0038**	**8.4**
Balance	**0.9089±0.0146**	32.2	0.8560±0.0326	19.8	0.8416±0.0155	9.3	0.9066±0.0035	**9.2**
Vehicle	**0.8294±0.0400**	25.7	0.7060±0.0468	22.4	-	-	0.7797±0.0112	**20.0**
Pageblocks	0.9496±0.0070	18.4	**0.9700±0.0055**	18.9	0.9067±0.0040	**16.2**	0.9550±0.0011	18.9
Texture	**0.9943±0.0030**	**15.7**	0.9520±0.0105	60.5	-	-	0.9856±0.0055	57.0
Magic	**0.8575±0.0062**	41.0	0.8540±0.0550	32.2	0.8160±0.0051	36.3	0.8533±0.0016	**18.9**
Penbased	0.9824±0.0051	**54.1**	0.9620±0.0056	119.2	-	-	**0.9828±0.0003**	98.7

Table 13
Comparison of EFSs on Real-World Binary Classification Problems

Algorithm	Electricity Pricing			Skin Segmentation		
	Acc	#(*Rules*)	t_{exe}	*Acc*	#(*Rules*)	t_{exe}
SAFL	**0.7430±0.0047**	5.9000±1.3703	**1.4409±0.1868**	**0.9850±0.0013**	21.2000±1.8738	**10.3593±0.4367**
eTS	0.4233±0.0038	15.1000±4.1218	285.9288±7.5234	0.9690±0.0109	11.3000±4.8086	1644.6519±50.5332
SAFIS	0.6166±0.0141	31.3000±12.7893	249.0422±168.1502	0.9698±0.0220	34.1000±5.5867	664.4516±107.0233
eClass0	0.5229±0.0339	8.4000±1.8379	1.6735±0.0747	0.8663±0.0808	**4.7000±3.0569**	10.9100±0.5149
eClass1	0.5979±0.1829	15.3000±3.7727	13.3720±1.4833	0.9668±0.0103	12.0000±3.8006	62.3002±7.9707
ALMMo	0.7254±0.0181	9.7000±2.1108	6.9957±0.4928	0.9282±0.0112	9.3000±1.9465	34.0865±0.8140
PALM	0.7337±0.0025	**1.0000±0.0000**	3.2944±0.0361	0.9439±0.0301	8.0000±11.5277	43.0871±53.4277

Table 14
Wilcoxon Test on Real-World Binary Classification Problems.

SAFL vs	Electricity Pricing		Skin Segmentation	
	p-value	*z-score*	*p-value*	*z-score*
eTS	0.0000	249.2328	0.0000	57.5203
SAFIS	0.0000	−75.4093	0.0000	9.0145
eClass0	0.0000	101.2712	0.0000	155.0987
eClass1	0.0000	173.1528	0.0000	59.2333
ALMMo	0.0000	−18.0122	0.0000	7.4397
PALM	0.0000	−30.8946	0.0000	27.9662

5.6. Further evaluations and discussions

Further to the above systematic evaluation, the robustness of the SAFL system is examined by adding Gaussian noise to the experimental data. In particular, this investigation is completed with six real-world benchmark datasets that have been previously used, namely, Mackey–Glass time series, nonlinear system identification cases 1 and 2, and three regression problems: Autos, Delta Ailerons, and California Housing. For each case study, 0 dB zero-mean Gaussian noise is added onto the input training samples. The prediction performance of SAFL in noisy environments is reported in Table 16, in comparison with the performances of eTS, SAFIS, ESAFIS and ALMMo under the same conditions. It can be observed from this table that SAFL exhibits top-level prediction precision and computational efficiency, outperforming the others, thereby demonstrating its strong robustness to noise.

Furthermore, as an approach designed for streaming data prediction, SAFL is tested by following the popular online learning evaluation method of test-then-train [15]. The experimental results on six benchmark regression problems obtained by SAFL and four alternative EFSs (eTS, SAFIS, ESAFIS and ALMMo) are tabulated in Table 17, from which it can be seen that SAFL surpasses its competitors in terms of both computational efficiency and prediction precision. This conforms to the general findings as reported in the preceding sections.

Last but not least, the test-then-train performance of SAFL is evaluated on the four non-stationary classification problems (namely, hyperplane, SEA, PMNIST and RMNIST) and compared with a variety of popular streaming data classification approaches. The comparative results are given in Table 18, where principle component analysis is used to reduce the dimensionality of PMNIST and RMNIST from 784 to 28 to facilitate computation. The reported results by SAFL is obtained as the average of 10 Monte Carlo simulations, and the results of PNN [39], DEN [48], HAT [41], pEnsemble+ [33], ADL [5], NADINE

Table 15
Results of Ablation Analysis

Dataset	Algorithm	*RMSE*	#(*Rules*)	t_{exe}
Mackey–Glass	SAFL	0.0238	20	0.1482
	SAFL-1	0.0412		0.4732
	SAFL-2	0.4426		0.2058
	SAFL-3	0.0488		0.4916
Nonlinear Case 1	SAFL	0.0052	12	0.2329
	SAFL-1	0.0277		0.5067
	SAFL-2	0.4806		0.2693
	SAFL-3	0.0311		0.5452
Nonlinear Case 2	SAFL	0.0955	9	0.1878
	SAFL-1	0.1517		0.4114
	SAFL-2	0.0883		0.2279
	SAFL-3	0.1462		0.4279
Delta Ailerons	SAFL	0.0492	36	0.2087
	SAFL-1	0.0493		0.7448
	SAFL-2	0.2989		0.2912
	SAFL-3	0.0497		0.9910
California Housing	SAFL	0.0692	26	0.6316
	SAFL-1	0.0706		2.1930
	SAFL-2	0.1336		0.8196
	SAFL-3	0.0724		2.7038

Table 16
Comparison on Regression Problems with Gaussian Noise.

Algorithm	**Mackey–Glass**			**Nonlinear System 1**		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	**0.2173±0.0026**	39.0400±2.8792	**0.1853±0.0104**	**0.3208±0.0094**	26.8800±2.1079	**0.2590±0.0167**
eTS	0.2209±0.0052	**9.7200±1.1733**	0.2320±0.0303	0.4099±0.0395	**9.7600±1.3317**	0.3837±0.0204
SAFIS	0.2175±0.0008	16.0400±1.6452	21.8230±0.3330	0.3226±0.0017	15.8400±0.3742	35.3043±0.8735
ESAFIS	0.2538±0.0616	34.5600±6.5579	8.1256±1.9529	0.5556±0.1724	51.8400±6.6124	16.7075±2.6211
ALMMo	0.2267±0.0301	131.1600±7.5481	78.5738±5.6601	0.3269±0.0230	46.3600±3.7403	41.1987±1.3102
Algorithm	**Nonlinear System 2**			**Autos**		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	**0.4217±0.0079**	14.2800±1.4000	**0.1913±0.0108**	0.2431±0.0207	54.0000±3.0551	**0.0093±0.0024**
eTS	0.4695±0.0185	10.3600±1.1860	0.3541±0.0169	0.1713±0.0087	**6.8000±2.3094**	0.0097±0.0037
SAFIS	0.4256±0.0008	**9.4400±0.8699**	39.4778±2.8726	**0.1708±0.0055**	7.6400±0.5686	0.6332±0.0648
ESAFIS	0.4838±0.1000	25.8800±4.2360	6.1900±1.7171	0.2465±0.0142	21.9200±6.8247	0.4481± 0.3314
ALMMo	0.4355± 0.0158	20.4400±1.8046	28.1313±1.7645	0.1987±0.0120	8.2000±0.9129	0.2256±0.0686
Algorithm	**Delta Ailerons**			**California Housing**		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	**0.0912±0.0005**	50.9200±3.2777	**0.2527±0.0138**	**0.1282±0.0004**	103.4400±5.8671	1.9832±0.0885
eTS	0.0919±0.0009	**9.5200±0.9626**	0.2567±0.0142	0.1289±0.0007	9.4000±0.9129	**0.8711±0.0461**
SAFIS	0.0913±0.0001	11.6400±0.8103	22.8132±1.0989	0.1284±0.0000	14.0400±0.6110	75.2098±1.3501
ESAFIS	0.0965±0.0047	20.8000±5.4467	5.1712±1.9948	0.1308±0.0006	22.6400±4.1721	30.2931±6.3529
ALMMo	0.0926±0.0006	31.3200±1.9732	25.1100±1.1788	0.1285±0.0004	**6.6000 ±1.1180**	26.6494±4.6956

[32] and MUSE-RNN [7] are obtained directly from [5,7,32]. Table 18 shows that SAFL outperforms the alternative approaches in terms of classification precision on all four problems.

In short, the systematic experimental studies (conducted over a variety of benchmark and real-world problems) collectively, and consistently, show that SAFL is superior to the state-of-the-art EFS algorithms, from the perspective of both prediction precision and computational efficiency. SAFL also has demonstrated its stability and robustness to noise. Although the number of fuzzy rules selected by SAFL may often be slightly greater than the average (e.g., it identifies 21 fuzzy rules on both problems of Mackey–Glass time series and S&P 500 closing price prediction, while SEFS works with 17 and 19 rules on these two problems, respectively), the training of SAFL consumes far less time. Finally, it is worth noting that all experimental results achieved by SAFL are obtained using the previously specified parameter settings, without any fine-tuning for optimisation. A significant space exists for strengthening SAFL's performance, via adjusting its three externally controlled parameters when given a particular domain problem. Similarly, the performance of a certain EFS may also be improved, subject to a number of factors in relation to the system specification, including: fuzzy rule types, structural evolving schemes, parameter learning mechanisms, membership function types, etc. Never-

Table 17
Comparison on Test-Then-Train Performance for Regression Problems.

Algorithm	Mackey–Glass			Nonlinear System 1			Nonlinear System 2		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	**0.0252**	20	**0.1631**	**0.0112**	12	**0.2303**	**0.0977**	9	**0.2104**
eTS	0.0620	**4**	1.3281	0.0141	**5**	35.9847	0.1341	**6**	35.9243
SAFIS	0.0551	6	2.9375	0.0437	10	3.4063	0.0997	11	1.4688
ESAFIS	0.0983	10	0.4336	0.0136	**5**	2.4219	0.1141	14	14.5781
ALMMo	0.0561	**4**	27.7500	0.0491	8	0.4219	0.1920	8	0.4955
Algorithm	**Autos**			**Delta Ailerons**			**California Housing**		
	RMSE	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}	*RMSE*	#(*Rules*)	t_{exe}
SAFL	0.0760	22	**0.0123**	**0.0503**	42	**0.4906**	**0.0694**	27	**1.2162**
eTS	0.0708	5	1.2385	0.0526	**4**	55.6398	0.0715	**3**	159.5139
SAFIS	0.0821	5	0.1406	0.0566	19	12.1563	0.0890	15	29.4688
ESAFIS	**0.0593**	**4**	0.1875	0.0503	15	26.6563	0.0698	6	37.7031
ALMMo	0.0739	8	0.0850	0.0535	10	0.6868	0.0782	10	1.8916

Table 18
Comparison on Test-Then-Train Performance for Classification Problems.

Algorithm	Hyperplane		SEA		PMNIST		RMNIST	
	Acc	t_{exe}	*Acc*	t_{exe}	*Acc*	t_{exe}	*Acc*	t_{exe}
SAFL	**0.9495±0.0008**	42.6242	**0.9794±0.0030**	**9.6245**	**0.9114±0.0007**	**94.1541**	**0.9117±0.0009**	**90.3826**
PNN	0.8507±0.0712	190.196	0.7527±0.0231	-	0.6442±0.0877	152.95	0.6094±0.1125	128.91
DEN	0.9183±0.0417	202.57	0.7995±0.1928	-	0.5208±0.2260	399.83	0.6148±0.2175	371.07
HAT	0.7790±0.1076	370.8	0.7465±0.0101	-	0.6300±0.1620	207.04	0.5400±0.0877	190.59
pEnsemble+	0.8760±0.0620	120	0.9200±0.0600	230	-	-	-	-
ADL	0.9233±0.0263	**21.51**	0.9213±0.0906	14	0.6840±0.2417	212	0.7290±0.0935	199
NADINE	-	-	0.9224±0.0640	15	0.7765±0.1509	202	0.7451±0.0750	192
MUSE-RNN	0.9264±0.0215	250.39	-	-	0.8387±0.1342	416.1	0.7627±0.0490	190.01

theless, all experimental studies reported in this work have been based on the same parameter settings wherever possible, thereby enabling fair comparison throughout.

6. Conclusion

This paper has presented a new approach that equips EFS with simplified inference and consequent parameter learning schemes. The resulting system, named SAFL performs prediction tasks (including regression and classification) with fewer fuzzy rules selected during the self-adaptive learning process. The stability of SAFL has been verified through both theoretical and empirical analysis. Systematic comparative investigations have demonstrated the efficacy and robustness of the proposed approach. There are a few considerations for future work. First, there are three parameters that require external control in order for the algorithm to function, which are currently determined empirically. It would be very interesting to develop certain data-driven, self-adjusting mechanisms to adapt these on the fly. Second, in the present implementation, stale fuzzy rules are detected and subsequently deleted on the basis of the average firing strengths within the current learning cycle. It would be useful to create alternative strategies to accomplish this more efficiently. Third, the premise and consequent parameters are not locally optimised in SAFL. It would be helpful to reinforce the prediction accuracy by integrating an optimisation means for such parameters with the system's learning process, and to observe how much additional computational cost this might incur. Finally, the computation cost of the RLS algorithm employed by most EFSs (including the present work) on problems involving high-dimensional data may become a significant challenge, due to the need of covariance matrix updating. This may restrict the applicability of EFSs to such problems like large-scale image classification and natural language processing, where DNNs generally work well ignoring the issue of interpretability. Whilst this challenge may be partially handled by exploiting dimensionality reduction, especially feature selection techniques (to retain attribute semantics) [22], it would be worthwhile to consider developing alternative consequent parameter learning algorithms such that EFSs can maintain its high computational efficiency on such problems.

CRediT authorship contribution statement

Xiaowei Gu: Formal analysis, Methodology, Visualization, Writing - original draft, Writing - review & editing. **Qiang Shen:** Methodology, Visualization, Writing - original draft, Writing - review & editing.

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.

References

[1] P. Angelov, D. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, IEEE Trans. Syst. Man, Cybern. - Part B Cybern. 34 (1) (2004) 484–498.
[2] P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in IEEE International Conference on Fuzzy Systems, 2005, pp. 1068–1073.
[3] P. Angelov, X. Gu, J. Principe, Autonomous learning multi-model systems from data streams, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 2213–2224.
[4] P. Angelov, X. Zhou, Evolving fuzzy-rule based classifiers from data streams, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1462–1474.
[5] A. Ashfahani, M. Pratama, Autonomous deep learning: continual learning approach for dynamic environments, in: SIAM International Conference on Data Mining, 2019, pp. 666–674.
[6] R. Bao, H. Rong, P. Angelov, B. Chen, P. Wong, Correntropy-based evolving fuzzy neural system, IEEE Trans. Fuzzy Syst. 26 (3) (2018) 1324–1338.
[7] M. Das, M. Pratama, S. Savitri, J. Zhang, MUSE-RNN: a multilayer self-evolving recurrent neural network for data stream classification, in: IEEE International Conference on Data Mining, 2019, pp. 110–119.
[8] J. de Jesus Rubio, P. Angelov, J. Pacheco, Uniformly stable backpropagation algorithm to train a feedforward neural network, IEEE Trans. Neural Networks 22 (3) (2011) 356–366.
[9] D. Dovzan, V. Logar, I. Skrjanc, Implementation of an evolving fuzzy model (eFuMo) in a monitoring system for a waste-water treatment process, IEEE Trans. Fuzzy Syst. 23 (5) (2015) 1761–1776.
[10] D. Dovzan, I. Skrjanc, Fuzzy space partitioning based on hyperplanes defined by eigenvectors for Takagi-Sugeno fuzzy model identification, IEEE Trans. Ind. Electron. 67 (6) (2020) 5144–5153.
[11] R. Duda, P. Hart, D. Stork, Pattern classification, 2nd ed., Wiley-Interscience, 2000.
[12] M. Fazzolari, R. Alcala, F. Herrera, A multi-objective evolutionary method for learning granularities based on fuzzy discretization to improve the accuracy-complexity trade-off of fuzzy rule-based classification systems: D-MOFARC algorithm, Appl. Soft Comput. 24 (2014) 470–481.
[13] S. Feng, C.L.P. Chen, L. Xu, Z. Liu, "On the accuracy-complexity trade-off of fuzzy broad learning system," IEEE Trans, Fuzzy Syst. (2020), https://doi.org/10.1109/tfuzz.2020.3009757.
[14] M. Ferdaus, M. Pratama, S. Anavatti, M. Garratt, PALM: an incremental construction of hyperplanes for data stream regression, IEEE Trans. Fuzzy Syst. 27 (11) (2019) 2115–2129.
[15] J. Gama, Knowledge discovery from data streams, CRC Press, 2010.
[16] J. Garibaldi, The need for fuzzy AI, IEEE/CAA J. Autom. Sin. 6 (3) (2019) 610–622.
[17] D. Ge and X. J. Zeng,"Learning evolving T-S fuzzy systems with both local and global accuracy - a local online optimization approach," Applied Soft Computing, vol. 86, pp. 795–810, 2018.
[18] D. Ge, X. Zeng, A self-evolving fuzzy system which learns dynamic threshold parameter by itself, IEEE Trans. Fuzzy Syst. 27 (8) (2019) 1625–1637.
[19] D. Ge and X. Zeng,"Learning data streams online- an evolving fuzzy system approach with self-learning/adaptive thresholds," Inf. Sci. (Ny)., vol. 507, pp. 172-184, 2020.
[20] X. Gu, Q. Shen, P. Angelov, Particle swarm optimized autonomous learning fuzzy system, IEEE Trans. Cybern. (2020), https://doi.org/10.1109/TCYB.2020.2967462.
[21] S. Jagannathan,"Control of a class of nonlinear discrete-time systems using multilayer neural networks," IEEE Trans. Neural Networks, vol. 12, no. 5, pp. 1113–1120, 2001.
[22] R. Jensen, Q. Shen, Computational intelligence and feature selection: rough and fuzzy approaches, IEEE Press and Wiley & Sons, 2008.
[23] N. 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.
[24] G. Leng, T. McGinnity, and G. Prasad,"An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network," Fuzzy Sets Syst., vol. 150, no. 2, pp. 211–243, 2005.
[25] F. Li, C. Shang, Y. Li, J. Yang, Q. Shen, Interpolation with just two nearest neighbouring weighted fuzzy rules, IEEE Trans. Fuzzy Syst. 28 (9) (2020) 2255–2262.
[26] Y. Li, H. Zhang, X. Xue, Y. Jiang, Q. Shen, Deep learning for remote sensing image classification: a survey, WIREs Data Min Knowl Discov y 8 (6) (2018) e1264.
[27] E.D. Lughofer, FLEXFIS: a robust incremental learning approach for evolving Takagi-Sugeno fuzzy models, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1393–1410.
[28] 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.
[29] M. Pratama, S. Anavatti, P. Angelov, E. Lughofer, PANFIS: a novel incremental learning machine, IEEE Trans. Neural Networks Learn. Syst. 25 (1) (2014) 55–68.
[30] M. Pratama, S. Anavatti, E. Lughofer, Genefis: toward an effective localist network, IEEE Trans. Fuzzy Syst. 22 (3) (2014) 547–562.
[31] M. Pratama, J. Lu, E. Lughofer, G. Zhang, M. Er, An incremental learning of concept drifts using evolving type-2 recurrent fuzzy neural networks, IEEE Trans. Fuzzy Syst. 25 (5) (2017) 1175–1192.
[32] M. Pratama, C. Za'in, A. Ashfahani, Y.S. Ong, W. Ding, Automatic construction of multi-layer perceptron network from streaming examples, in: International Conference on Information and Knowledge Management, 2019, pp. 1171–1180.
[33] M. Pratama, E. Dimla, T. Tjahjowidodo, W. Pedrycz, E. Lughofer, Online tool condition monitoring based on parsimonious ensemble+, IEEE Trans. Cybern. 50 (2) (2020) 664–677.
[34] H. Rong, P. Angelov, X. Gu, J. Bai, Stability of evolving fuzzy systems based on data clouds, IEEE Trans. Fuzzy Syst. 26 (5) (2018) 2774–2784.
[35] H. Rong, G. Huang, N. Sundararajan, P. Saratchandran, Online sequential fuzzy extreme learning machine for function approximation and classification problems, IEEE Trans. Syst. Man, Cybern.- Part B Cybern. 39 (4) (2009) 1067–1072.
[36] H. Rong, N. Sundararajan, G. Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction, Fuzzy Sets Syst. 157 (9) (2006) 1260–1275.
[37] H. Rong, N. Sundararajan, G. Huang, G. Zhao, Extended sequential adaptive fuzzy inference system for classification problems, Evol. Syst. 2 (2) (2011) 71–82.
[38] H. Rong, Z. Yang, P. Wong, "Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system," IEEE Trans, Fuzzy Syst. (2019), https://doi.org/10.1109/TFUZZ.2019.2931871.
[39] A. Rusu, N. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu and R. Hadsell, "Progressive neural networks," arXiv Prepr. arXiv1606.04671, 2016.
[40] S. Samanta, M. Pratama, S. Sundaram, A novel spatio-temporal fuzzy inference system (SPATFIS) and its stability analysis, Inf. Sci. (Ny) 505 (2019) 84–99.

[41] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: in International Conference on Machine Learning, 2018, pp. 4548–4557.
[42] I. Skrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey, Inf. Sci. (Ny) 490 (2019) 344–368.
[43] I. Skrjanc, Cluster-volume-based merging approach for incrementally evolving fuzzy Gaussian clustering-eGAUSS+, IEEE Trans. Fuzzy Syst. 28 (9) (2020) 2222–2231.
[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (56) (2014) 1929–1958.
[45] K. Subramanian, S. Suresh, A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system, Appl. Soft Comput. J. 12 (11) (2012) 3603–3614.
[46] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man. Cybern. 15 (1) (1985) 116–132.
[47] L. Wang, J. Mendel, Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, IEEE Trans. Neural Networks 3 (5) (1992) 807–814.
[48] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," arXiv Prepr. arXiv1708.01547, 2017.
[49] Y. Zhang, H. Ishibuchi, S. Wang, Deep Takagi-Sugeno-Kang fuzzy classifier with shared linguistic fuzzy rules, IEEE Trans. Fuzzy Syst. 26 (3) (2018) 1535–1549.