Information Sciences 634 (2023) 382–399

Contents lists available at ScienceDirect

Information Sciences

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

Self-adaptive fuzzy learning ensemble systems with dimensionality compression from data streams

Xiaowei Gu

School of Computing, University of Kent, Canterbury CT2 7NZ, UK

A R T I C L E I N F O

Keywords:
Data stream
Dimensionality compression
Ensemble learning
Evolving fuzzy system
Prediction

A B S T R A C T

Ensemble learning is a widely used methodology to build powerful predictors from multiple individual weaker ones. However, the vast majority of ensemble learning models are designed for offline application scenarios, the use of evolving fuzzy systems in ensemble learning for online learning from data streams has not been sufficiently explored, yet. In this paper, a novel self-adaptive fuzzy learning ensemble system is introduced for data stream prediction. The proposed ensemble system employs the very sparse random projection technique to compress the consequent parts of the learned fuzzy rules by individual base models to a more compressed form, thereby reducing redundant information and improving computational efficiency. To improve the overall prediction performance, a dynamical base model pruning scheme is introduced to the proposed ensemble system together with a novel inferencing scheme, such that less accurate base models will be removed from the ensemble structure at each learning cycle automatically and only these more accurate ones will be involved in joint decision-making. Numerical examples based on a wide range of benchmark datasets demonstrate the stronger prediction performance of the proposed ensemble system over the state-of-the-art alternatives.

1. Introduction

Ensemble learning is a powerful machine learning methodology to construct a stronger predictor by combining multiple individual weaker predictors [1]. The main motivation of using ensemble models comes from the no-free-lunch theorem [2]. According to the theorem, there is no single predictor that is appropriate for all tasks. However, by weighting and integrating multiple individual predictors into one, one can obtain a new predictor that outperforms each of these individual ones [3]. Ensemble learning has been a widely researched topic of great importance in the recent decades and has been applied successfully in various real-world applications to tackle complex challenging classification and regression problems.

Currently, bagging [4] and boosting [5] are the two mainstream approaches for creating ensemble models. As a parallel ensemble mechanism, bagging samples the training dataset with replacement into multiple subsets and uses each of these subsets to train a base model (predictor). These trained predictors are then combined via majority voting. Boosting, on the other hand, is a sequential ensemble mechanism. It trains a series of individual base models consecutively with various distributed training data such that each predictor complements its predecessors. Then, these individual predictors are combined by weighted majority voting with more weights assigned to the predictors that perform better on the training set. Compared with bagging, boosting gives more focus to harder samples, leading to better prediction performance. However, bagging is more robust to noisy training samples and is less likely to be

E-mail address: x.gu@kent.ac.uk.

https://doi.org/10.1016/j.ins.2023.03.123
Received 5 September 2022; Received in revised form 9 March 2023; Accepted 19 March 2023
Available online 23 March 2023

overfitting [6].

It is well known that to construct a good ensemble predictor, the base models need to be both accurate and diverse. The vast majority of existing works on ensemble learning employ mainstream machine learning models [3], such as support vector machine (SVM) [7], artificial neural network (ANN), decision tree (DT) [8] and k-nearest neighbour (KNN) [9], etc., as the base models, and have reported great performance. However, these mainstream models are mostly designed to learn from static data, some of which are also widely criticized as being opaque (i.e., SVM and ANN) [10], and ensemble models constructed by mainstream models are often limited to offline application scenarios.

Evolving fuzzy systems (EFSs) [11–13] are a special class of fuzzy systems designed for data stream processing. They are capable of self-developing and self-evolving the system structure and parameters from data streams online, and are the prominent methodology used for real-time non-stationary problem approximation [14]. Classical EFSs include, but are not limited to, dynamic evolving neural-fuzzy inference system (DENFIS) [11], evolving Takagi-Sugeno (eTS) fuzzy model [12], evolving fuzzy rule-based (eClass0 and eClass1) classifiers [13], sequential adaptive fuzzy inference system (SAFIS) [15], flexible fuzzy inference system (FLEXFIS) [16], parsimonious learning machine (PALM) [17], self-evolving fuzzy system (SEFS) [18], recursive maximum correntropy-based evolving fuzzy system (RMCEFS) [19], evolving fuzzy system with self-learning/adaptive thresholds (EFS-SLAT) [20], etc.

In recent years, there have been a few works proposed to construct ensemble models using fuzzy systems. For instance, an AdaBoost-based ensemble system consisted of relational fuzzy systems is proposed in [21], and this work is later extended in [22] by using logical-type neuro-fuzzy classifiers as base models. A fuzzy rule-based ensemble system utilizing attribute regrouping is proposed in [23], where each base model is trained on a unique lower dimensional projection of data. The idea of utilizing EFSs as base components of ensemble systems for data stream mining is firstly proposed in [24], and an eClass0-based ensemble (eEnsemble) classifier is implemented to demonstrate the concept. Parsimonious ensemble (pENsemble) proposed in [25] is an EFS-based ensemble classifier with an evolving ensemble structure that can automatically add or prune base models to self-adapt to the changes in underlying patterns of data streams. pENsemble is also equipped with a dynamic online feature selection scheme that can detect and remove redundant input attributes online. A deep evolving fuzzy neural network (DEVFNN) is introduced in [26] under the stacked generalization principle via the feature augmentation concept. DEVFNN is composed of multiple generalized Takagi-Sugeno-Kang fuzzy systems stacked one by one in layers, where the output of the previous layer is used for augmenting the input of the next layer. As such, the base models in DEVFNN are interconnected through augmentation of feature space. Inspired by the multi-layered structure of deep neural networks (DNNs), a multi-layered ensemble evolving fuzzy inference system (MEEFIS) with first-order EFSs as building blocks is introduced in [27]. Benefited from its deeper ensemble structure, MEEFIS can learn multi-layered distributed representations from data and demonstrates stronger capability to handle large-scale, non-stationary, complex problems. A fuzzily weighted adaptive boosting (FWADABoost) method designed for creating stronger zero-order ensemble EFS is introduced in [28]. A unique feature of FWADABoost is the use of confidence scores produced by EFSs in both weight updating and ensemble output generation to create more precise classification boundaries. In [29], an online bagging-based ensemble fuzzy classifier with an autonomous pruning scheme, which removes base models with higher prediction errors, is proposed. An online sequential ensemble of fuzzy predictors is proposed in [30] for chunk-wise data stream learning. In this framework, data streams are received in the form of chunks, and a new fuzzy predictor is initialized and added to the ensemble when a new data chunk is available. Other fuzzy system-based ensemble models worth mentioning include selected ensemble of fuzzy inference systems (SENFIS) [31], self-organizing fuzzy inference ensemble (SOFEnsemble) system [32], parsimonious ensemble (pENsemble+) [33], etc. Despite of the great potential exhibited by existing works, the use of EFSs in ensemble learning, especially in online application scenarios, has not been sufficiently explored, yet [28,29].

In this paper, a novel self-adaptive fuzzy learning ensemble (SAFLE) system is introduced for streaming data prediction. The SAFLE system employs a group of self-adaptive fuzzy learning (SAFL) as its base models. SAFL is a recently introduced EFS with a unique activation control scheme that dynamically selects a small number of more activated fuzzy rules by the current input at each learning cycle for approximation reasoning and parameter updating [14]. Thanks to the removal of irrelevant information, the computational complexity of SAFL is greatly reduced and its prediction accuracy is improved at the same time.

To increase the diversity between the base models of SAFLE whilst maintaining their prediction performances, the very sparse random projection (VSRP) technique [34] is employed to compress the dimensionality of the consequent parameters of SAFL. Hence, the new variant is named as SAFL-VSRP. Compared with the original SAFL, SAFL-VSRP associates a unique VSRP matrix to each individual rule within its rule base, such that the consequent parts of the fuzzy rules are compressed to a tighter form with redundant information between existing rules being removed from the rule base. In doing so, the computational efficiency of individual SAFL-VSRP systems is greatly increased, especially when the dimensionality of data is high.

Due to the compressed dimensionality by VSRP, some of the base models may inevitably perform worse due to the loss of information despite of the increase of diversity between them. To remove the error-prone base models and improve the overall prediction accuracy of joint decision-making, a dynamic base model pruning scheme is introduced to the proposed SAFLE system. At each learning cycle, the pruning scheme will identify the base models that tend to make more prediction errors and remove them from the ensemble structure. To maintain the original ensemble size, new SAFL-VSRP systems will be initialized to fill in the empty spaces left by the pruned base models. In addition to the pruning scheme, SAFLE is also equipped with a novel inferencing scheme that selects only these base models with less prediction errors to perform joint decision-making, thereby improving its overall prediction performance.

Different from existing fuzzy ensemble approaches in the literature, the proposed SAFLE system has a unique ensemble architecture that each individual SAFL-VSRP system is associated with a pair of training and validation data pools to dynamically maintain two small non-overlapping subsets of historical data in the system memory. At each learning cycle, data stored in the validation pools will be used to evaluate the performances of the individual base models. The out-of-sample prediction accuracy on validation data serves as the main criterion for SAFLE to identify these weaker base models that need to be pruned in order to achieve greater overall prediction performance. Once a particular base model is pruned due to its poor prediction performance, the data stored in the associated training data pool will be combined with the most recent training data received at the current learning cycle and utilized to initialize a new SAFL-VSRP system as a replacement. The use of both new and historical data enables the newly created base models to capture the latest data patterns whilst reflecting historical changes, with the potential of greatly improving the prediction performance of SAFLE, especially in the case of regular and cyclic drifts [30].

To summarize, key features of the proposed SAFLE system include:

1. An ensemble architecture to store small amounts of historical data for measuring the out-of-sample prediction performance of individual base models and creating new base models;

2. The utilization of VSRP matrices in the consequent parts of fuzzy rules to improve the diversity and computational efficiency of individual base models;

3. A dynamic pruning scheme to identify error-prone base models and replace them with new base models initialized on a combination of new and historical data for greater prediction performance.

4. A fuzzy inferencing scheme to utilize only elite base models in joint decision-making with the respective weights computed based on their out-of-sample prediction performances.

The remainder of this paper is organized as follows. Section 2 gives the problem definition. Technical details of the proposed SAFL-VSRP are described in Section 3. The general architecture, online learning policy and joint decision-making policy of the proposed SAFLE system are presented in Section 4. Numerical examples are given in Section 5 as a proof of concept, and this paper is concluded by Section 6.

2. Problem definition

The proposed SAFLE system is designed to autonomously learn from data streams on a chunk-by-chunk basis for prediction in online application scenarios.

Let $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_k, \cdots\}$ be a particular data stream in a M-dimensional data space with $\{\boldsymbol{y}\} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \cdots, \boldsymbol{y}_k, \cdots\}$ being the corresponding target/real outputs, where $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \cdots, x_{k,M}]^T$; $\boldsymbol{y}_k = \left[y_{k,1}, y_{k,2}, \cdots, y_{k,K}\right]^T$ is the corresponding K-dimensional target output vector of $\boldsymbol{x}_k$, and; the subscript k denotes the time instance at which $\boldsymbol{x}_k$ is observed. It is also assumed that samples of the data stream $\{\boldsymbol{x}\}$ continuously arrive in chunks denoted as $\{\boldsymbol{x}\}_1, \{\boldsymbol{x}\}_2, \cdots, \{\boldsymbol{x}\}_i, \cdots$, where $\{\boldsymbol{x}\}_i = \{\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \cdots, \boldsymbol{x}_{i,L}\}$ is the i th data chunk with the cardinality, L. Note that cardinalities of different data chunks may vary in real applications.

Due to the inherent nonstationary nature of data streams, the underlying data patterns can change at any occasion. In other words, concept drifts may happen frequently and the data distributions may vary from time to time, namely, $P_i(\boldsymbol{x}) \neq P_j(\boldsymbol{x})$ and/or $P_i(\boldsymbol{y}|\boldsymbol{x}) \neq P_j(\boldsymbol{y}|\boldsymbol{x})\ \forall i \neq j$ [26]. The existence of potential concept drifts poses a significant challenge to existing machine learning models for data stream processing because the models are required to be able to precisely approximate the problem in real time whilst having the capability to react to the rapid changes of data patterns.

To effectively handle concept drifts, the proposed SAFLE system continuously monitors the prediction performance of each individual base model in its ensemble framework at each learning cycle and replaces these error-prone base models with new ones to quickly self-adapt to new data patterns. There are two unique aspects that distinguish the proposed SAFLE from existing works in similar areas.

1. The prediction performance of each individual base model is evaluated at each learning cycle based on validation samples dynamically maintained in the associated validation data pool, which are independent from data samples used for model learning. Thus, the obtained out-of-sample prediction error serves as an objective measure of the predictive capability and generalization ability of the base model.

2. Once a weaker base model is pruned due to the poor prediction performance, a new base model is initialized at the next learning cycle by utilizing both the current training data and a small selected set of historical training data stored in the associated training data pool. In this way, the new base model not only captures the latest data patterns, but also gains the knowledge of historical data patterns, resulting in greater prediction performance.

In addition, to increase the diversity between individual base models and lower their computational costs, SAFLE introduces a unique randomly generated VSRP matrix to each individual fuzzy rule learned by its ensemble components from data. These VSRP matrices compress the input data into a more compact form and effectively reduce the computational complexity of consequent parameter updating by downsizing the covariance matrices, thereby improving the overall computational efficiency of the ensemble model.

In the next two sections, technical details of the proposed SAFL-VSRP and SAFLE models are presented.

3. Proposed SAFL-VSRP

3.1. Architecture

The architecture of SAFL-VSRP is presented in Fig. 1 [14], where one can see that a multiple-input multiple-output (MIMO) SAFL-VSRP system is composed of N IF-THEN fuzzy rules in the form of Eq. (1) ($n = 1, 2, \cdots, N$; N is the number of rules in the fuzzy rule base)

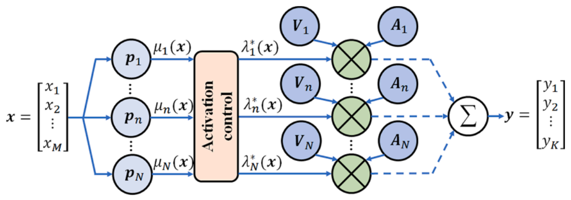

Fig. 1. Architecture of SAFL-VSRP [14].

[34]. Such IF-THEN rules are a variant of the prototype-based IF-THEN rules used in [14] with the additional VSRP matrices in the consequent parts to project the input vectors to a lower dimensional latent space.

$$\boldsymbol{R}_n: \quad IF\,(\boldsymbol{x} \sim \boldsymbol{p}_n) \quad THEN\left(\boldsymbol{y}_n = \boldsymbol{A}_n\left[1, \frac{1}{\sqrt{M_c}}\boldsymbol{x}^T\boldsymbol{V}_n\right]^T\right) \tag{1}$$

where "∼" denotes similarity; $\boldsymbol{x} = [x_1, x_2, \cdots, x_M]^T$ is the $M \times 1$ dimensional input vector; $\boldsymbol{y}_n = \left[y_{n,1}, y_{n,2}, \cdots, y_{n,K}\right]^T$ is the $K \times 1$ dimensional output vector of $\boldsymbol{R}_n$; M is the dimensionality of the input vector; K is the dimensionality of the output vector; $M_c = \lfloor \varepsilon M \rfloor$ is the dimensionality of the input vector after compression; ε is the compression ratio, $0 < \varepsilon \leq 1$; $\boldsymbol{p}_n = \left[p_{n,1}, p_{n,2}, \cdots, p_{n,M}\right]^T$ is the prototype of $\boldsymbol{R}_n$; $\boldsymbol{A}_n = \left[\boldsymbol{a}_{n,1}, \boldsymbol{a}_{n,2}, \cdots, \boldsymbol{a}_{n,K}\right]^T$ is the $K \times (M_c + 1)$ dimensional consequent parameter matrix of $\boldsymbol{R}_n$; $\boldsymbol{a}_{n,k} = \left[a_{n,k,0}, a_{n,k,1}, \cdots, a_{n,k,M_c}\right]^T$; $y_{n,k} = \boldsymbol{a}_{n,k}^T\left[1, \boldsymbol{x}^T\boldsymbol{V}_n\right]^T$; $\boldsymbol{V}_n = \left[v_{n,l,j}\right]_{j=1:M_c}^{l=1:M}$ is the $M \times M_c$ dimensional VSRP matrix associated with $\boldsymbol{R}_n$; and $v_{n,l,j}$ is the element at the l th row and j th column of $\boldsymbol{V}_n$ ($l = 1, 2, \cdots, M$; $j = 1, 2, \cdots, M_c$), which is randomly generated by the following distribution as specified by [34]:

$$v_{n,l,j} = \sqrt{\alpha} \bullet \begin{cases} 1, & \text{with probability } \dfrac{1}{2\alpha} \\ 0, & \text{with probability } 1 - \dfrac{1}{\alpha} \\ -1, & \text{with probability } \dfrac{1}{2\alpha} \end{cases} \tag{2}$$

Here α is set as $\sqrt{M}$, following [34]. It has been shown in previous studies that, compared with using a typical random matrix consisted of entries with a standard value range of $[0,1]$ or $[-1,1]$, using a VSRP matrix for dimensionality compression can achieve a significant α-fold speedup with marginal loss in accuracy [34].

Note that ε is an externally controlled parameter controlling the dimensionality of the input vector after compression. The smaller ε, the lower dimensional the input vector becomes. A small ε can greatly improve the computational efficiency of the online learning process of SAFL-VSRP, but would inevitably decrease its prediction accuracy due to the loss of information.

Compared with conventional first-order EFSs, the computational efficiency of SAFL-VSRP is greatly improved thanks to 1) the unique dynamic rule selection scheme, which isolates the less activated fuzzy rules from fuzzy inferencing and consequent parameter updating at each learning cycle and 2) the dimensionality compression scheme, which reduces the computational complexity of covariance matrix updating greatly. This rule selection scheme also helps SAFL-VSRP to achieve greater prediction accuracy, whilst guaranteeing the stability of the system [14]. The online learning and decision-making policies of SAFL-VSRP are detailed as follows.

3.2. Online learning policy

The online learning process of SAFL-VSRP is composed of the following five stages.

Stage 1. System initialization. Given a new input sample, $\boldsymbol{x}_i$, the first fuzzy rule is initialized if $\boldsymbol{x}_i$ is the very first sample, namely, $i = 1$. Firstly, the global parameters of the system are initialized using Eq. (3).

$$\boldsymbol{\upsilon} \leftarrow \boldsymbol{x}_i; \quad \chi \leftarrow \|\boldsymbol{x}_i\|^2 \tag{3}$$

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

The first cluster, $\mathbf{C}_N$ is built with $\boldsymbol{x}_i$ being the prototype ($N \leftarrow 1$):

$$\mathbf{C}_N \leftarrow \{\boldsymbol{x}_i\}; \boldsymbol{p}_N \leftarrow \boldsymbol{x}_i; S_N \leftarrow 1; \boldsymbol{c}_N \leftarrow \boldsymbol{x}_i; X_N \leftarrow \|\boldsymbol{x}_i\|^2; \; I_N \leftarrow i; \; \Lambda_N \leftarrow 1 \tag{4}$$

where S_N is support of $\mathbf{C}_N$ (the number of data samples associated with $\mathbf{C}_N$); $\boldsymbol{c}_N$ is the arithmetic mean of these data samples; X_N is the corresponding average squared Euclidean norm; I_N denotes the time instance at which $\mathbf{C}_N$ is initialized and Λ_N is the accumulated firing strength starting from I_N to the current time instance.

The first fuzzy rule, $\boldsymbol{R}_N$ is then initialized based on $\mathbf{C}_N$ with $\boldsymbol{p}_N$ being the antecedent part in the form of Eq. (1). The consequent part of $\boldsymbol{R}_N$ is set by Eq. (5) as follows.

$$\boldsymbol{A}_N \leftarrow \mathbf{0}_{K\times(M_c+1)};\ \boldsymbol{\theta}_N \leftarrow \Omega_o \mathbf{I}_{(M_c+1)\times(M_c+1)} \tag{5}$$

where $\mathbf{0}_{K\times(M_c+1)}$ is the $K\times(M_c+1)$ dimensional zero matrix; $\boldsymbol{\theta}_N$ is the covariance matrix; $\mathbf{I}_{(M_c+1)\times(M_c+1)}$ is a $(M_c+1)\times(M_c+1)$ dimensional identity matrix, and; Ω_o is a constant for covariance matrix initialization used by standard RLS algorithms [11,12]. In this study, $\Omega_o = 10000$. The VSPR matrix associated with $\boldsymbol{R}_N$, denoted as $\boldsymbol{V}_N$ is then created by Eq. (2).

Note that each time a new fuzzy rule is created, a new VSRP matrix will be randomly generated and assigned to the new fuzzy rule for dimensionality compression. Thus, each fuzzy rule within the rule base of SAFL-VSRP will be associated with a unique VSRP matrix. However, once a VSRP matrix is created, it will not be updated anymore during its lifespan until the corresponding fuzzy rule is removed by Condition (2). The main reason for fixing these VSRP matrices is because updating these randomly generated VSRP matrices during online learning process will significantly decrease the computational efficiency of SAFL-VSRP due to the higher computational complexity of data projection and the extra computational costs for updating these matrices. In addition, SAFLE is equipped with a dynamic pruning scheme to remove these poorly performed base models (which will be detailed in Section 4.2). The proposed pruning scheme naturally favours these base models with high-quality VSRP matrices for more precise ensemble decision making.

If $\boldsymbol{x}_i$ is not the first data sample, namely, $i > 1$, the learning process enters Stage 2 directly.

Stage 2. Activation control. Given a new data sample, $\boldsymbol{x}_i$ $(i \leftarrow i+1)$, the global parameters, $\boldsymbol{\upsilon}$ and χ are updated firstly using Eq. (6).

$$\boldsymbol{\upsilon} \leftarrow \boldsymbol{\upsilon} + \frac{\boldsymbol{x}_i - \boldsymbol{\upsilon}}{i};\quad \chi \leftarrow \chi + \frac{\|\boldsymbol{x}_i\|^2 - \chi}{i} \tag{6}$$

The firing strength of each individual fuzzy rule, $\boldsymbol{R}_n$ $(n = 1,2,\cdots,N)$ is calculated as follows:

$$\mu_n(\boldsymbol{x}_i) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{p}_n\|^2}{\sigma_n^2}} \tag{7}$$

where $\sigma_n = \sqrt{\frac{(\chi - \|\boldsymbol{\upsilon}\|^2) + (X_n - \|\boldsymbol{c}_n\|^2)}{2}}$ is the radius of area of influence around $\boldsymbol{p}_n$.

Rule activations of the N fuzzy rules are ranked in descending order as $\overline{\mu}_1(\boldsymbol{x}_i) \geq \overline{\mu}_2(\boldsymbol{x}_i) \geq \cdots \geq \overline{\mu}_N(\boldsymbol{x}_i)$, and the most activated N^* rules by $\boldsymbol{x}_i$ are identified using Eq. (8).

$$N_i = \underset{n=1,2,\cdots,N}{\operatorname{argmin}} \left(\sum\nolimits_{j=1}^{n} \overline{\mu}_j(\boldsymbol{x}_i) > \rho_o \bullet \sum_{j=1}^{N} \overline{\mu}_j(\boldsymbol{x}_i) \right) \tag{8}$$

where $0 \leq \rho_o \leq 1$; there is always $N_i \leq \rho_o N$. It can be observed from Eq. (8) that if ρ_o is equal to 1, all the fuzzy rules will be selected, whilst only the most activated rule will be selected if ρ_o is equal to 0. In this study, $\rho_o = 0.5$ is recommended, unless specific domain knowledge is available for determining its value [14].

It is generally a common practice for the vast majority of first-order EFSs [11–13,15] that all the fuzzy rules within the rule base will be updated in response to the current input sample and involved in fuzzy inferencing. However, updating the consequent parts of these less activated rules only introduces tiny changes to the consequent parameters with no improvement in the overall system performance because of the marginal firing strength values, but may significantly increase the computational complexity especially if the dimensionality of data is high. As the final output of an EFS is typically computed as the weighted sum of outputs of these individual rules given a particular input sample, involving these fuzzy rules with marginal firing strengths in producing the system output also does not improve the overall prediction performance. On the other hand, by selecting only the more activated rules in response to the current input for consequent parameter updating and fuzzy inferencing, the computational complexity of consequent parameter updating can be effectively reduced and overfitting can be avoided [14].

It is also worth noting that, rather than being a hardcoded crisp threshold, ρ_o specifies the ratio between the firing strengths of the selected fuzzy rules and that of the entire rule base. Based on this ratio, Eq. (8) identifies these relatively more activated rules than other candidates from the rule base in response to the input sample, which is more robust, flexible and guaranteed to be meaningful [14].

Then, the normalized firing strengths of the fuzzy rules are calculated by Eq. (9) $(n = 1,2,\cdots,N)$.

$$\lambda_n^*(\boldsymbol{x}_i) = \begin{cases} \dfrac{\mu_n(\boldsymbol{x}_i)}{\sum_{j=1}^{N_i} \overline{\mu}_j(\boldsymbol{x}_i)}, & \mu_n(\boldsymbol{x}_i) \geq \overline{\mu}_{N_i}(\boldsymbol{x}_i) \\ 0, & else \end{cases} \tag{9}$$

For these less activated rules, their normalized rule activations are set to be 0. These rules will not contribute to fuzzy inferencing, nor be updated at the current learning cycle.

Stage 3. System structure updating. For the current data sample, $\boldsymbol{x}_i$, Condition (1) is examined to select whether it represents a novel data pattern and has the potential to initialize a fuzzy rule:

$$Cond.1:\quad if\ (\overline{\mu}_1(\boldsymbol{x}_i) < \mu_0)\quad then\ (\boldsymbol{x}_i \text{ initializes a fuzzy rule}) \tag{10}$$

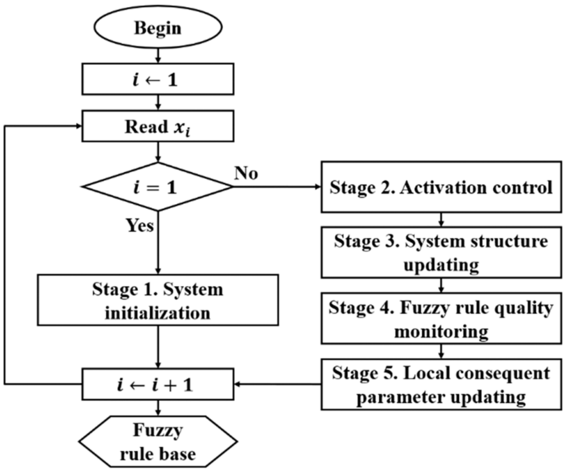
Begin
$i \leftarrow 1$
Read x_i
$i = 1$
No
Yes
Stage 2. Activation control
Stage 3. System structure updating
Stage 4. Fuzzy rule quality monitoring
Stage 5. Local consequent parameter updating
Stage 1. System initialization
$i \leftarrow i + 1$
Fuzzy rule base

Fig. 2. Diagram of SAFL-VSRP learning process.

where $\mu_0 = e^{-1}$, corresponding to the so-called "one sigma" rule [35].

If Condition (1) is satisfied, a new cluster, $\mathbf{C}_N$ ($N \leftarrow N + 1$) is initialized by $\boldsymbol{x}_i$ using Eq. (4), and a new fuzzy rule $\boldsymbol{R}_N$ is initialized with $\boldsymbol{x}_i$ being its prototype in the form of Eq. (1). The consequent part of the newly added fuzzy rule is set by Eq. (11) with the corresponding VSRP matrix, $\boldsymbol{V}_N$ created by Eq. (2).

$$\boldsymbol{A}_N \leftarrow \frac{1}{N-1}\sum_{n=1}^{N-1}\boldsymbol{A}_n; \boldsymbol{\theta}_N \leftarrow \Omega_o \mathbf{I}_{(M_c+1)\times(M_c+1)} \tag{11}$$

As shown in Eq. (11), the consequent parameter matrix, $\boldsymbol{A}_N$ of the new rule $\boldsymbol{R}_N$ is initialized as the average of the consequent parameter matrices of all other fuzzy rules within the rule base. This is a standard strategy adopted by many EFSs [14,27] in the literature. However, one may also consider alternative strategies, for example, initializing the consequent parameters of the new rule as a matrix of all zeros [16,36].

Otherwise, $\boldsymbol{x}_i$ is used for updating the parameters of the cluster that produces the greatest firing strength, $\overline{\mu}_1(\boldsymbol{x}_i)$ with regard to $\boldsymbol{x}_i$, denoted as $\mathbf{C}_{n^*}$:

$$\mathbf{C}_{n^*} \leftarrow \mathbf{C}_{n^*} \cup \{\boldsymbol{x}_i\}; S_{n^*} \leftarrow S_{n^*} + 1; \boldsymbol{c}_{n^*} \leftarrow \boldsymbol{c}_{n^*} + \frac{\boldsymbol{x}_i - \boldsymbol{c}_{n^*}}{S_{n^*}}; X_{n^*} \leftarrow X_{n^*} + \frac{\|\boldsymbol{x}_i\|^2 - X_{n^*}}{S_{n^*}} \tag{12}$$

where $\boldsymbol{c}_{n^*}$, X_{n^*} and S_{n^*} are the arithmetic mean, average squared Euclidean norm and support of $\mathbf{C}_{n^*}$, respectively. After that, the firing strength $\mu_{n^*}(\boldsymbol{x}_i)$ is updated using Eq. (7) to reflect the latest changes. The normalized firing strengths $\mu_{n^*}(\boldsymbol{x}_i)$ are also updated accordingly using Eq. (9).

Stage 4. Fuzzy rule quality monitoring. SAFL-VSRP removes these stale fuzzy rules from the rule base on the accumulated firing strengths to improve the prediction accuracy. To do so, the accumulated firing strength of each fuzzy rule is firstly updated by Eq. (13) ($n = 1, 2, \cdots, N$):

$$\Lambda_n \leftarrow \Lambda_n + \frac{\mu_n(\boldsymbol{x}_i) - \Lambda_n}{I_n - i} \tag{13}$$

Then, Condition (2) is used to identify these stale fuzzy rules:

$$Cond.2: \quad if\ (\Lambda_n < \Lambda_0) \quad then\ (\boldsymbol{R}_n \text{ and } \mathbf{C}_n \text{ are removed}) \tag{14}$$

where Λ_0 is a constant determining the tolerance of SAFL-VSRP towards stale fuzzy rules, and $\Lambda_0 = 0.05$ [14]. Once Condition (2) is satisfied for $\boldsymbol{R}_n$, it is removed from the rule base together with the corresponding cluster, $\mathbf{C}_n$.

Stage 5. Local consequent parameter updating. The consequent parameters of these activated fuzzy rules, (namely, $\lambda_n^*(\boldsymbol{x}_i) \neq 0$) at the current learning cycle are updated using the fuzzily weighted recursive least square (FWRLS) algorithm [12] as follows ($n = 1, 2, \cdots, N$).

$$\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}_n - \frac{\lambda_n^*(\boldsymbol{x}_i)\boldsymbol{\theta}_n \overline{\boldsymbol{x}}_{i,n}\overline{\boldsymbol{x}}_{i,n}^T\boldsymbol{\theta}_n}{1 + \lambda_n^*(\boldsymbol{x}_i)\overline{\boldsymbol{x}}_{i,n}^T\boldsymbol{\theta}_n\overline{\boldsymbol{x}}_{i,n}} \tag{15}$$

$$\boldsymbol{A}_n \leftarrow \boldsymbol{A}_n - \lambda_n^*(\boldsymbol{x}_i)\left(\boldsymbol{A}_n\overline{\boldsymbol{x}}_{i,n} - \boldsymbol{y}_i\right)\overline{\boldsymbol{x}}_{i,n}^T\boldsymbol{\theta}_n \tag{16}$$

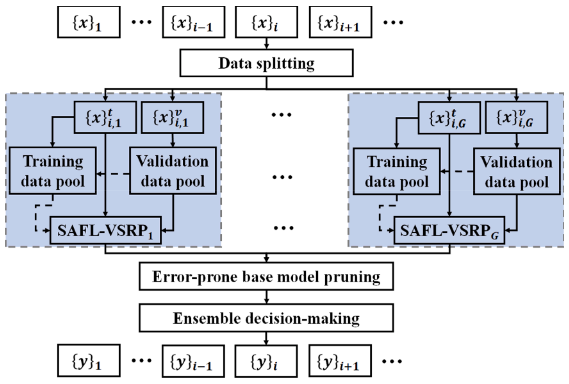

Fig. 3. Architecture of SAFLE system.

where $\boldsymbol{\theta}_n$ and $\boldsymbol{A}_n$ are the covariance matrix and consequent parameters of the n th rule, $\boldsymbol{R}_n$ if it is activated; $\overline{\boldsymbol{x}}_{i,n} = \left[1, \frac{1}{\sqrt{M_c}}\boldsymbol{x}_i^T\boldsymbol{V}_n\right]^T$. For these fuzzy rules that are not activated, namely, $\lambda_n^*(\boldsymbol{x}_i) = 0$, their consequent parameters are kept the same for the next learning cycle. The main reason for using FWRLS in SAFL-VSRP is because FWRLS [12] is the most popular approach for consequent parameter learning used by a wide variety of first-order EFSs, including SAFL [14]. However, one may also consider alternative approaches instead, such as weighted recursive least square [11], extended Kalman filter [15], adaptive maximum correntropy extended Kalman filter [37], etc.

After that, the SAFL-VSRP learning process goes back to Stage 2, and a new learning cycle begins once the next data sample is available. The diagram of the learning process of SAFL-VSRP is given by Fig. 2 for better illustration.

3.3. *Decision-making policy*

Given a new data sample $\boldsymbol{x}$, the system output of SAFL-VSRP is produced as the weighted sum of outputs of all the activated fuzzy rules by $\boldsymbol{x}$ [14]:

$$\widehat{y} = f(\boldsymbol{x}) = \sum_{n=1}^{N} \lambda_n^*(\boldsymbol{x})\boldsymbol{A}_n\left[1, \frac{1}{\sqrt{M_c}}\boldsymbol{x}^T\boldsymbol{V}_n\right]^T \tag{17}$$

where $\lambda_n^*(\boldsymbol{x})$ is the normalized firing strength of $\boldsymbol{R}_n$ calculated by Eq. (9). Not that if $\lambda_n^*(\boldsymbol{x}) = 0$, $\boldsymbol{R}_n$ will not contribute to the final system output of SAFL-VSRP in response to the current input, $\boldsymbol{x}$.

3.4. *Computational complexity analysis*

The computational complexity of the proposed SAFL-VSRP is analysed in this section. Since SAFL-VSRP learns from data streams sample-by-sample, the analysis is conducted at the i th time instance at which $\boldsymbol{x}_i$ is observed.

Stage 1 considers system initialization only and is not performed for any $i > 1$. Hence, the computational complexity of Stage 1 is negligible within the overall learning process. Stage 2 is to produce the rule activations and select out the candidate rules from the rule base for system structure updating (Stage 3). The complexity for updating the global parameters, $\boldsymbol{v}$ and χ is $O(M)$, for calculating the firing strengths of the fuzzy rules is $O(MN)$, for identifying these activated rules is $O(N)$ and for calculating the normalized firing strengths is $O(N_i)$. Stage 3 is for system structure updating and the complexity for adding or updating a rule is $O(M)$. Stage 4 identifies and removes stale rules, and the complexity of that is $O(N)$. Stage 5 concerns updating the consequent parameters of the candidate fuzzy rules. The complexity for dimensionality compression is $O\left(M_c\sqrt{M}N_i\right)$ and for updating consequent parameters is $O\left((M_c+1)^2N_i\right)$. Hence, the computational complexity of a particular learning cycle of SAFL is $O\left(\left((M_c+1)^2 + M_c\sqrt{M}\right)N_i\right)$. The overall computational complexity of the entire learning process given a total of i input samples is $O\left(\left((M_c+1)^2 + M_c\sqrt{M}\right)\sum_{j=1}^{i}N_i\right)$.

Compared with the overall complexity of SAFL, which is $O\left((M+1)^2\sum_{j=1}^{i}N_i\right)$ [14], it can be concluded that the computational complexity of the learning process of SAFL-VSRP is much lower than the original SAFL given a small compression ratio.

4. Proposed SAFLE

4.1. *Architecture*

The architecture of SAFLE is depicted in Fig. 3. According to Fig. 3, SAFLE is composed of *i)* a data splitting module, *ii)* G base models (namely, SAFL-VSRP systems), *iii)* an error-prone model pruning module, and *iv)* an ensemble decision-making module. There are also a pair of training and validation data pools associated with each individual base model to store a small selected group of historical training and validation data.

4.2. Online chunk-wise learning policy

SAFLE learns from data streams on a chunk-by-chunk basis. The learning procedure of SAFLE consists of the following three stages.

Stage 1. Random data splitting. Once a new data chunk $\{x\}_i$ with the chunk size L is available, for each base model (assuming the g th one, $g = 1,2,\cdots,G$), the data splitting module will firstly divide the current chunk $\{x\}_i$ into a pair of non-overlapping training and validation sets in a purely random manner at a prefixed ratio, γ $(0 < \gamma < 1)$. The obtained training and validation sets are denoted as $\{x\}_{i,g}^{t}$ and $\{x\}_{i,g}^{v}$. The respective sizes of $\{x\}_{i,g}^{t}$ and $\{x\}_{i,g}^{v}$ are γL and $(\gamma - 1)L$, and there are $\{x\}_{i,g}^{t} \cup \{x\}_{i,g}^{v} = \{x\}_i$ and $\{x\}_{i,g}^{t} \cap \{x\}_{i,g}^{v} = \varnothing$. After that, the training set $\{x\}_{i,g}^{t}$ is passed to the g th base model and the associated training data pool, denoted by $\boldsymbol{X}_g^t$. The validation set $\{x\}_{i,g}^{v}$ is passed to the corresponding validation data pool, denoted by $\boldsymbol{X}_g^v$. Note that the data splitting module will repeatedly divide the data chunk for G base models such that each base model will receive a different pair of training and validation sets. This effectively enhances the diversity between different base models since each base model utilizes a different (almost independent) subset of data for training.

Stage 2. Base model learning. If $\{x\}_i$ is the very first data chunk, namely, $i = 1$, the training and validation data pools will be set as: $\boldsymbol{X}_g^t \leftarrow \{x\}_{i,g}^{t}$ and $\boldsymbol{X}_g^v \leftarrow \{x\}_{i,g}^{v}$. To guarantee the memory efficiency of the SAFLE system, the sizes of the training and validation data pools will be fixed as γL and $(\gamma - 1)L$ after initialization. Then, the g th base model, denoted as $f_g(\boldsymbol{x})$ $(g = 1,2,\cdots,G)$ will be created from $\{x\}_{i,g}^{t}$.

Otherwise, namely, $i > 1$, the training and validation data pools will combine the stored historical samples with the current ones, namely, $\boldsymbol{X}_g^t \leftarrow \boldsymbol{X}_g^t \cup \{x\}_{i,g}^{t}$ and $\boldsymbol{X}_g^v \leftarrow \boldsymbol{X}_g^v \cup \{x\}_{i,g}^{v}$. The two pools temporarily become oversized as their sizes are doubled temporarily, namely, $2\gamma L$ and $2(\gamma - 1)L$. After that, the g th base model will be updated with $\{x\}_{i,g}^{t}$. However, if the g th base model, $f_g(\boldsymbol{x})$ has been pruned in the previous learning cycle, a new base model will be initialized with the oversized training data pool, $\boldsymbol{X}_g^t$ to replace the old one. The main reason for utilizing both the historical and current training data in creating new base model is to help the new model rapidly capture the latest data patterns whilst having a good understanding about the historical patterns. Note that both the initialization and updating processes follow the same online learning procedure described in Section 3.2.

Stage 3. Error-prone base model pruning. After the base models have been created/updated, the model pruning module will monitor their prediction performances and remove these weaker (namely, less accurate) ones to maintain a healthy ensemble framework. To do so, the prediction error of each base model (assuming the g th one) will be evaluated based on data samples stored within the oversized validation data pool, $\boldsymbol{X}_g^v$. The prediction error of the g th base model, denoted as e_g $(g = 1,2,\cdots,G)$ will be measured in terms of classification error if SAFLE is tackling classification problems:

$$e_g = \frac{1}{2(\gamma - 1)L} \sum_{\boldsymbol{x} \in \boldsymbol{X}_g^v} \mathbb{I}(\widehat{y} \neq y) \tag{18a}$$

where $\widehat{y}$ is the predicted label of $\boldsymbol{x}$ by the g th base model; y is the corresponding true label of $\boldsymbol{x}$ $(\boldsymbol{x} \in \boldsymbol{X}_g^v)$;

Alternatively, if SAFLE is applied to regression problems, e_g will be measured in terms of non-dimensional error index (*NDEI*):

$$e_g = \min\left(1, \frac{1}{\sigma_y} \sqrt{\frac{1}{2(\gamma - 1)L} \sum_{\boldsymbol{x} \in \boldsymbol{X}_g^v} \|\widehat{y} - y\|^2}\right) \tag{18b}$$

where y is the corresponding real output of $\boldsymbol{x}$ $(\boldsymbol{x} \in \boldsymbol{X}_g^v)$; σ_y is the standard deviation of the real outputs associated with $\boldsymbol{X}_g^v$. Here e_g is capped at 1 in Eq. (18b) to ensure that the value range of e_g is always between 0 and 1. One may further consider to normalize the prediction errors of individual base models to the value range of $[0, 1]$ in case that all the base models achieve the error upper limit 1, but that would be extremely unlikely to happen.

The base models with poorer prediction performances are identified by Condition (3).

$$Cond.3: \quad if\ \left(e_g > (\mu_e + \Delta_e)\right) \quad then\ \left(f_g(\boldsymbol{x}) \text{ is pruned}\right) \tag{19}$$

where μ_e and Δ_e are the respective mean and standard deviation of the prediction errors of all the base models. Condition (3) is also based on the so-called “one sigma” rule [35], allowing SAFLE to identify and prune these error-prone models agilely. The prediction errors of these base models are measured based on both the current and historical validation samples that are not presented to the base models for parameter learning. Hence, the obtained performance measures are highly objective and reliable, providing a good indication of how well each base model self-adapts to the latest changes of data patterns and responses to familiar data patterns.

Condition (3) (Eq. (19)) has a similar form to the pruning condition proposed in [29] and both conditions aim to identify and prune these weaker base models. However, their main difference lies in the way that the prediction errors of individual base models are measured. In the proposed SAFLE system, the prediction error of a base model is evaluated based on validation samples, which are independent from training samples used for model updating. In contrast, in [29], the prediction errors of base models are measured based on the accumulated training errors between predicted outputs with respect to the input training samples and the corresponding real outputs. Thus, Condition (3) used in SAFLE gives a more objective evaluation on the out-of-sample prediction performance of individual base models. Importantly, the measured prediction errors on validation data reflect the generalization abilities of the base models and can help SAFLE identify those overfitted base models.

After these error-prone base models are removed from the ensemble framework, each validation data pool, X_g^v ($g = 1, 2, \cdots, G$) is firstly downsized to the original size of $(\gamma - 1)L$ via randomly sampling. Next, the redundant validation samples squeezed out of the validation data pools are merged into the corresponding training data pools. The training data pools, X_g^t ($g = 1, 2, \cdots, G$) are then downsized to the original sizes of γL via randomly sampling, again. In this way, the information carried by redundant validation samples is preserved (partially) in the training data pools for creating new models instead of being discarded completely. At the end of the current learning cycle, the SAFLE learning process goes back to Stage 1 to learn from the next available data chunk ($i \leftarrow i + 1$).

For clarify, the SAFLE learning process is summarized by the following pseudo code. It is worth noting that data samples in the training pools are used to initialize new base models only after weaker base models are pruned at the current learning cycle. Otherwise, the training data pools only collect data samples without participating in the learning process. Data samples in the validation pools, on the other hand, will be used at each learning cycle to help SAFLE identify weaker base models.

Algorithm 1: SAFLE learning process.

```
$i \leftarrow 1$;
**while** (a new data chunk $\{x\}_i$ is available) **do**
**for** $g = 1$ to $G$ **do**
randomly split $\{x\}_i$ to $\{x\}_{i,g}^t$ and $\{x\}_{i,g}^v$ at the ratio of $\gamma$;
**if** ($i = 1$) **then**
initialize training data pool: $X_g^t \leftarrow \{x\}_{i,g}^t$;
initialize validation data pool: $X_g^v \leftarrow \{x\}_{i,g}^v$;
build $f_g(x)$ from $\{x\}_{i,g}^t$;
**else**
expand training data pool: $X_g^t \leftarrow X_g^t \cup \{x\}_{i,g}^t$;
expand validation data pool: $X_g^v \leftarrow X_g^v \cup \{x\}_{i,g}^v$;
**if** ($f_g(x)$ was pruned at the *(i-1)*th learning cycle) **then**
initialize $f_g(x)$ from $X_g^t$;
**else**
update $f_g(x)$ from $\{x\}_{i,g}^t$;
**end if**
**end if**
evaluate the prediction error $e_g$ of $f_g(x)$ on $X_g^v$;
**if** (Condition (3) is satisfied) **then**
prune $f_g(x)$;
**end if**
downsize $X_g^t$ and $X_g^v$ via random sampling;
**end for**
$i \leftarrow i + 1$;
**end while**
```

4.3. Ensemble decision-making policy

During decision making, SAFLE will select a sub-group of base models based on their prediction performances on the validation data to perform fuzzy inference. Only these base models that satisfy Condition (4) will contribute to decision making:

$$Cond.4: \quad if \left(e_g \le \left(\mu_e^{'} + \Delta_e^{'}\right)\right) \quad then \left(f_g(x) \,\text{is involved in decision making}\right) \tag{20}$$

where $\mu_e^{'}$ and $\Delta_e^{'}$ are the mean and standard deviation of the prediction errors of all the base models within the ensemble framework, respectively. Condition (4) is almost identical to Condition (3). However, since these weaker base models have already been pruned by Condition (3) during the learning process, only these stronger base models are kept in the ensemble framework. These selected candidates by Condition (4) are even more accurate and reliable. Utilizing only these elite models for joint decision making can lead to more accurate predictions compared with involving all the base models regardless of their respective performances.

After these elite base models, re-denoted as $f_1^*(x), f_2^*(x), \cdots, f_{N^*}^*(x)$ ($N^* \le N$) are identified by Condition (4), given a new data sample x, the corresponding output of SAFLE is obtained as the weighted sum of the outputs of the selected base models:

$$\hat{y} = \frac{1}{\sum_{g=1}^{N^*}\left(1 - e_g^*\right)} \sum_{g=1}^{N^*} \left(1 - e_g^*\right) f_g^*(x) \tag{21}$$

where e_g^* is the prediction error of $f_g^*(x)$; $g = 1, 2, \cdots, N^*$. It can be seen from Eq. (21) that these base models with less prediction error are given greater weights in the joint decision making.

Table 1
Key information of classification problems.

Dataset		Abbr.	#(Attributes)	#(Samples)	#(Classes)	Characteristics
Occupancy Detection	Training	OD	5	8143	2	Stationary
	Testing			12,417		
Multiple Features		MF	649	2000	10	Stationary
Pen-Based Recognition of Handwritten Digits	Training	PR	16	7494	10	Stationary
	Testing			3498		
Optical Recognition of Handwritten Digits	Training	OR	64	3823	10	Stationary
	Testing			1797		
Abalone		AB	8	4177	3	Stationary
Image Segmentation	Training	IS	19	420	7	Stationary
	Testing			2100		
MAGIC Gamma Telescope		MA	10	19,020	2	Stationary
Pima Indians diabetes		PI	8	768	2	Stationary
Page-blocks		PB	10	5472	5	Stationary
Spambase		SP	57	4601	2	Stationary
Semeion handwritten digit		SH	256	1593	10	Stationary
Texture		TE	40	5500	11	Stationary
Phishing websites		PW	30	11,055	2	Stationary
German credit		GC	24	1000	2	Stationary
Electricity pricing		EP	8	45,312	2	Non-stationary
Skin segmentation		SS	3	245,057	2	Non-stationary
SUSY		SU	18	5,000,000	2	Non-stationary
Hyperplane		HP	4	120,000	2	Non-stationary
SEA		SE	3	200,000	2	Non-stationary
MNIST permutations		PM	784	70,000	10	Non-stationary
MNIST rotations		RM	784	65,000	10	Non-stationary

One may also notice that the ensemble decision-making scheme used by SAFLE is similar to the one used in [30] in the sense that both schemes compute the weighted sums of the outputs of the base models to produce the final outputs. However, SAFLE utilizes only the elite base models in joint decision-making, whilst the scheme proposed in [30] involves all the base models. The two schemes also compute the weights of base models differently. In [30], higher weights are given to these base models that perform better on training samples. The model weights are further adjusted with respect to each input sample during the decision-making process with extra weights allocated to the more confident base models. In contrast, SAFLE assigns higher weights to these base models with greater out-of-sample prediction performances on validation data. Thus, SAFLE has a lower risk of overfitting.

5. Experimental investigation

In this section, numerical examples are conducted for evaluating the performance of the proposed SAFLE system. The algorithms are developed using MATLABR2021b platform and experiments are performed on a desktop with dual core i7 processor 3.80 GHz $\times$ 2 and 32.0 GB RAM.

5.1. Dataset description

In this work, a wide range of popular benchmark problems are used to evaluate the performance of the proposed SAFLE system, which include 21 classification problems, one high frequency trading (HFT) prediction problem and one S&P500 closing price prediction problem. In particular, the 21 classification problems investigated include 14 classical benchmarks[1], three real-world ones (electricity pricing[2], skin segmentation[3] and SUSY[4]), two synthetic ones (hyperplane and SEA[5]) and two visual ones (MNIST permutations and rotations[6]). Key information of 21 classification problems is summarized in Table 1. Note that to facilitate computation, the original attributes of two visual classification problems are compressed by principal component analysis with the dimensionality reduced from 784 to 28.

The HFT and S&P500 problems are detailed as follows.

1. HFT: This dataset contains tick-by-tick data on all NASDAQ, NYSE and AMEX securities from 1998 to the present with 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 a total of 19,144 samples involved in this problem. In this case study, two specific experiments are conducted as follows [19,36]:

[1] Available at: https://archive.ics.uci.edu/ml/datasets.php.
[2] Available at: https://moa.cms.waikato.ac.nz/datasets/.
[3] Available at: https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation.
[4] Available at: https://archive.ics.uci.edu/ml/datasets/SUSY.
[5] Available at: https://scikit-multiflow.github.io/.
[6] Available at: https://nlp.stanford.edu/projects/mer/.

Table 2
Classification error (*Err*) comparison between SAFLE and alternative classification approaches.

Algorithm	OD	MF	PR	OR	AB	IS	MA
SAFLE	**0.0361**	**0.0221**	0.0405	0.0262	**0.4533**	0.1211	0.1894
SAFL	0.0534	0.0256	0.0356	0.0263	**0.4518**	0.1096	**0.1706**
SVM	0.0405	0.0569	**0.0229**	0.0287	0.4642	**0.0967**	0.1722
KNN	0.0420	0.0697	0.0240	**0.0206**	0.4762	0.1562	0.1960
EC	0.0385	0.0987	0.0852	0.1358	0.4622	0.1300	0.1908
SEQ	0.4811	0.0252	0.0489	0.0451	0.5479	0.1176	0.2236
SDKNN	0.2649	0.0235	0.0469	0.0373	0.5494	0.1619	0.2212
ELM	**0.0105**	0.0219	**0.0221**	0.0837	0.6649	0.8570	0.4515
PNN	0.1233	0.0538	0.0380	**0.0200**	0.4729	0.1967	0.1901
RF	**0.0401**	**0.0228**	0.1424	0.0879	**0.4506**	0.0436	**0.1586**
SAMME	0.0760	0.0744	0.2240	0.2006	0.4589	**0.0915**	0.1726
XGBoost	0.0485	0.0233	0.0383	0.0417	0.4671	**0.0610**	**0.1249**
eEnsemble	0.1252	0.1821	0.1921	0.1445	0.5096	0.2500	0.3429
FWADABoost	0.0444	0.0607	**0.0239**	**0.0223**	0.4627	0.2331	0.2309
SOFEnsemble	0.0554	**0.0219**	0.0563	0.0891	0.4689	0.2219	0.2720
Algorithm	PI	PB	SP	SH	TE	PW	GC
SAFLE	**0.2413**	0.0420	0.1069	**0.0783**	**0.0045**	0.0669	**0.2400**
SAFL	0.2460	0.0445	0.1024	0.0961	**0.0048**	0.0657	0.2586
SVM	0.2872	0.0533	0.1927	**0.0795**	0.0137	**0.0483**	0.2888
KNN	0.2751	0.0490	0.2178	0.1230	0.0219	0.0697	0.3088
EC	0.2955	**0.0308**	0.1417	0.2944	0.0338	0.0539	0.2940
SEQ	0.4053	0.0577	0.1147	0.1065	0.0186	0.0530	0.3384
SDKNN	0.4042	0.0384	0.1135	0.1828	0.0272	0.0906	0.2962
ELM	0.3342	0.0959	0.1184	0.5573	0.0901	0.0890	**0.2188**
PNN	0.2649	0.1781	0.1102	0.1111	0.0205	**0.0342**	0.3358
RF	**0.2412**	**0.0325**	**0.0755**	0.1779	0.1361	0.0713	0.2626
SAMME	**0.2602**	0.0338	**0.0956**	0.3017	0.1619	0.0716	0.2764
XGBoost	0.2614	**0.0275**	**0.0513**	0.1095	0.0276	**0.0522**	**0.2492**
eEnsemble	0.3813	0.1912	0.3125	0.2834	0.1756	0.2174	0.3724
FWADABoost	0.3239	0.0444	0.1767	**0.0949**	**0.0119**	0.0588	0.2930
SOFEnsemble	0.2797	0.0489	0.2279	0.1013	0.0206	0.0601	0.3106

- o Using the current four prices to predict the high price eight steps ahead:

$$\widehat{x}_{k+8,2} = f\left(\left[x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}\right]^T\right) \tag{22}$$

- o Using the current four prices to predict the high price 24 steps ahead:

$$\widehat{x}_{k+24,2} = f\left(\left[x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}\right]^T\right) \tag{23}$$

Following the common practice [19,36], the four prices have been standardized online. The prequential test-then-train protocol is employed for experimental simulation.

2. S&P500: This dataset contains daily closing prices of S&P500 ranging from 03/01/1950 to 12/03/2009, 14,893 samples in total. This dataset is further expanded with its flipped time series, and thus, 29,786 samples after augmentation [20]. Following the common practice, all data has been normalized to the range of [0, 1], with the first half of the augmented dataset used for training and the second half used for online testing under the prequential test-then-train protocol. The regression model is defined as follows.

$$\widehat{x}_{k+1} = f\left(\left[x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1}, x_k\right]^T\right) \tag{24}$$

Unless specifically declared otherwise, the proposed SAFLE system is composed of 10 base models, namely, $G = 10$, and the compression ratio ε, chunk size L and split ratio γ are set as $\varepsilon = 0.75$, $L = 2000$ and $\gamma = 0.8$. During the experiments, if the amount of remaining unprocessed data samples is less than L, all the remaining samples will be included in the last chunk. It has to be stressed that this is a general setting, and the best performing parameters can be different from problem to problem and require prior knowledge to be determined a priori.

To better understand the influences of the externally controlled parameters, G, ε, L and γ on the performance of SAFLE model, sensitivity analysis is performed based on three regression problems, and the results are presented in Appendix A. In addition, an ablation analysis is conducted based on four synthetic classification problems and presented in Appendix B to demonstrate that the use of dimensionality compression can largely improve the computational efficiency of SAFLE without sacrificing prediction accuracy.

All the reported results in this study are obtained as the average of 10 Monte Carlo experiments to allow a certain degree of randomness.

Table 3
Average classification errors (*Err*) and training time consumptions (t_{exe}) of the classification approaches over 14 benchmark problems.

Algorithm	*Err*	t_{exe}
SAFLE	**0.1192**	36.24
SAFL	**0.1208**	8.83
SVM	0.1318	0.34
KNN	0.1464	**0.01**
EC	0.1632	705.22
SEQ	0.1845	0.68
SDKNN	0.1756	0.15
ELM	0.2582	**0.04**
PNN	0.1535	**0.01**
RF	0.1388	0.16
SAMME	0.1785	1.12
XGBoost	**0.1131**	0.41
eEnsemble	0.2629	0.32
FWADABoost	0.1487	3.58
SOFEnsemble	0.1596	0.34

5.2. Performance evaluation

In the first numerical example, the performance of the proposed SAFLE system is tested on 14 classical benchmark classification problems. During the experiments, except for OD, PR, OR and IS datasets, which have prefixed training–testing splits, for the other 10 datasets, 50% of the data samples are randomly selected for training and the remaining samples are used for testing. The classification results obtained by SAFLE on the 14 datasets are tabulated in Table 2 in the form of classification error (*Err*).

For better evaluation, the following classification approaches are used for comparison:

o SAFL system [14];
o SVM [7];
o KNN classifier [9];
o EigenClass (EC) classifier [38];
o Sequence (SEQ) classifier [39];
o Sequence-dictionary-based KNN (SDKNN) classifier [39];
o Extreme learning machine (ELM) [40];
o Probabilistic neural network (PNN) [41];
o Random forest (RF) [42];
o Stagewise additive modelling using a multi-class exponential loss function (SAMME) [43];
o XGBoost classifier [8];
o eEnsemble classifier [25];
o FWADABoost classifier [28], and;
o Self-organizing fuzzy inference ensemble (SOFEnsemble) system [32].

In running the experiments, SAFL follows the original setting as [14]. SVM uses Gaussian kernel, and the kernel width is determined automatically using a heuristic procedure. Both KNN and SDKNN use $k = 5$. EC considers the first five eigenvalues for classification. SEQ follows the same setting as [39]. The maximum number of neurons for ELM is set as 200. The standard deviation of PNN is set as $\frac{\delta_x}{\sqrt{2}}$ (δ_x is the standard deviation of training data). RF, SAMME, XGBoost, eEnsemble, SOFEnsemble and FWADABoost are six ensemble classification algorithms. SAMME and XGBoot use DT as base classifiers, same as RF. The maximum split of DT used by RF, SAMME and XGBoost is set as 20, and the number of base classifiers in the ensemble models is set as 50. $\eta = 0.3$ is used for XGBoost. eEnsemble, FWADABoost and SOFEnsemble are ensemble classifiers based on EFSs, and they use the same parameter settings as [25,28,32], respectively. The classification performances of the comparative approaches are also reported in Table 2, where the top three results achieved on each dataset are highlighted in bold. For visual clarity, the average classification errors (*Err*) of the classification approaches over the 14 datasets are given in Table 3, and the respective average training time consumptions (t_{exe}, in seconds) are reported in Table 3 as well (the top three results are in bold).

It can be seen from Tables 2 and 3 that the proposed SAFLE system is able to achieve the second highest classification accuracy ($1 - Err$) over the 14 classical benchmark problems among the classification approaches involved in the example. Despite that the overall classification error of XGBoost is lower than SAFLE, SAFLE outperforms XGBoost on eight out of the 14 datasets, which include OD, MF, OR, AB, PI, SH, TE and GC. This performance comparison demonstrates the very strong classification performance of the proposed SAFLE system. On the other hand, one may notice that SAFLE requires more computational resources than the majority of the classification approaches involved in comparison, though the dimensionality compression technique has been employed. This is due to the online recursively updating of covariance matrices associated with the fuzzy rules, which is less computationally efficient when the dimensionality of data is high. Nevertheless, SAFLE uses 3.62 s in average to train a base model (composed of 10 base models in this

Table 4
Classification performance comparison on two real-world classification problems.

Dataset	Algorithm	*Err*	#(Rules)	t_{exe}
EP	**SAFLE**	0.2647 ± 0.0078	59.60 ± 3.84	13.85 ± 0.23
	SAFL	**0.2570 ± 0.0047**	5.90 ± 1.37	1.44 ± 0.19
	eTS	0.5767 ± 0.0038	15.1000 ± 4.12	285.93 ± 7.52
	SAFIS	0.3834 ± 0.0141	31.30 ± 12.79	249.04 ± 168.15
	eClass0	0.4771 ± 0.0339	8.40 ± 1.84	**1.67 ± 0.07**
	eClass1	0.4021 ± 0.1829	15.30 ± 3.77	13.37 ± 1.48
	PALM	0.2663 ± 0.0025	**1.00 ± 0.00**	3.29 ± 0.04
	SEFIS	0.4141 ± 0.0253	25.00 ± 0.00	9.48 ± 0.24
	eEnsemble	0.4168 ± 0.0059	176.70 ± 17.13	2.28 ± 0.32
SS	**SAFLE**	**0.0142 ± 0.0014**	234.90 ± 8.02	94.79 ± 1.75
	SAFL	0.0150 ± 0.0013	21.20 ± 1.87	10.36 ± 0.44
	eTS	0.0310 ± 0.0109	11.30 ± 4.81	1644.65 ± 50.53
	SAFIS	0.0302 ± 0.0220	34.10 ± 5.59	664.45 ± 107.01
	eClass0	0.1337 ± 0.0808	4.70 ± 3.06	10.91 ± 0.51
	eClass1	0.0332 ± 0.0103	12.00 ± 3.80	62.30 ± 7.97
	PALM	0.0561 ± 0.0301	**8.00 ± 11.53**	43.09 ± 53.43
	SEFIS	0.1573 ± 0.0431	24.70 ± 0.67	47.06 ± 0.58
	eEnsemble	0.0683 ± 0.0016	109.30 ± 2.21	**10.46 ± 0.47**

case), which is less than half of the time needed for training a single SAFL model. Although the experiments are conducted on a single desktop, the computational efficiency of SAFLE can be greatly improved in practice by using distributed computing facilities for parallelization.

Next, the performance of SAFLE is tested on two non-stationary real-world classification problems, namely, EP and SS. For the two datasets, 80% of the samples are randomly selected for training and the remaining 20% for testing. Performance of SAFLE is reported in Table 4, in terms of *Err*, number of rules, #(Rules) and t_{exe}. In this example, the following EFSs are employed for performance comparison:

o SAFL system [14];
o eTS system [12];
o SAFIS [15];
o eClass0 classifier [13];
o eClass1 classifier [13];
o PALM [17];
o statistically evolving fuzzy inference system (SEFIS) [37], and;
o eEnsemble classifier [25].

In this experiments, the results of eTS, SAFIS, eClass0, eClass1 and PALM are obtained from [14] directly. For SEFIS, its externally controlled parameters are determined as: $K = 0.5$, $\delta_1 = 0.5$, $\delta_2 = 0.5$ and $p_0 = 2$. eEnsemble uses the same setting as [25]. The classification performances of the comparative EFSs on the two benchmark datasets are also presented in Table 4, where the best result on each dataset is highlighted in bold. It can be observed from Table 4 that SAFLE outperforms all the EFS-based comparative approaches on the two problems (only slightly worse than SAFL on EP), showing its strong capability in handling non-stationary problems.

Then, the test-then-train performance of the proposed SAFLE system is evaluated on five large-scale non-stationary classification problems, including SU, HP, SE, PM and RM. As aforementioned, to facilitate computation, principal component analysis is used to reduce the dimensionality of PM and RM from 784 to 28. The performance of SAFLE is compared with the following state-of-the-art classification approaches designed for large-scale non-stationary data streams:

o progressive neural network (ProgNN) [44];
o dynamically expandable network (DEN) [45];
o online multiclass boosting (OMCBoosting) [46];
o hard attention to the task (HAT) classifier [47];
o pEnsemble [25];
o pEnsemble+ [33];
o autonomous deep learning (ADL) classifier [48];
o neural network with dynamically evolved capacity (NADINE) [49], and;
o multilayer self-evolving recurrent neural network (MUSE-RNN) [50].

The performance comparison results are given in Table 5, where the results of ProgNN, DEN, OMCBoosting, HAT, pEnsemble+, ADL, NADINE and MUSE-RNN are obtained from [49,50] directly (the best results are in bold).

Finally, the prediction performance of SAFLE is further evaluated on HFT and S&P500 regression problems by following the test-

Table 5
Performance comparison on large-scale non-stationary data stream classification problems.

Dataset	Algorithm	*Err*	t_{exe}
SU	**SAFLE**	**0.2083 ± 0.0001**	85.7 K
	ProgNN	0.3106 ± 0.0408	345 K
	DEN	0.3685 ± 0.1006	8 K
	OMCBoosting	0.2287 ± 0.0143	14 K
	HAT	0.2615 ± 0.0318	16 K
	pEnsemble	0.2556 ± 0.0240	14 K
	pEnsemble+	0.2301 ± 0.0460	35 K
	ADL	0.2174 ± 0.0280	2.5 K
	NADINE	0.2197 ± 0.0300	**1.5 K**
	MUSE-RNN	0.2186 ± 0.0160	21 K
HP	**SAFLE**	**0.0194 0.0014**	0.4 K
	ProgNN	0.1493 ± 0.0712	0.2 K
	DEN	0.0817 ± 0.0417	0.2 K
	OMCBoosting	0.1382 ± 0.0373	0.1 K
	HAT	0.2210 ± 0.1076	3.7 K
	pEnsemble	0.0820 ± 0.0190	68
	pEnsemble+	0.1240 ± 0.0620	0.2 K
	ADL	0.0767 ± 0.0263	**22**
	NADINE	–	–
	MUSE-RNN	0.0736 ± 0.0215	0.3 K
SE	**SAFLE**	**0.0220 ± 0.0013**	0.1 K
	ProgNN	0.1513 ± 0.0652	0.2 K
	DEN	0.2005 ± 0.1928	0.2 K
	OMCBoosting	0.1322 ± 0.0385	77
	HAT	0.2535 ± 0.1010	0.3 K
	pEnsemble	0.0800 ± 0.0570	0.2 K
	pEnsemble+	0.9800 ± 0.0600	0.2 K
	ADL	0.0718 ± 0.0579	18
	NADINE	0.0776 ± 0.0640	**15**
	MUSE-RNN	0.0763 ± 0.0611	0.1 K
PM	**SAFLE**	**0.0825 ± 0.0002**	5.8 K
	ProgNN	0.3558 ± 0.0877	**0.2 K**
	DEN	0.4792 ± 0.2260	0.4 K
	OMCBoosting	0.6442 ± 0.2051	5 K
	HAT	0.4036 ± 0.1888	**0.2 K**
	pEnsemble	–	–
	pEnsemble+	–	–
	ADL	0.3160 ± 0.2417	**0.2 K**
	NADINE	0.2235 ± 0.1509	**0.2 K**
	MUSE-RNN	0.1613 ± 0.1342	0.4 K
RM	**SAFLE**	**0.0860 ± 0.0007**	10.5 K
	ProgNN	0.4381 ± 0.1094	0.1 K
	DEN	0.3852 ± 0.2173	0.4 K
	OMCBoosting	0.7393 ± 0.0580	5 K
	HAT	0.3548 ± 0.1133	0.2 K
	pEnsemble	–	–
	pEnsemble+	–	–
	ADL	0.2710 ± 0.0935	0.2 K
	NADINE	0.2549 ± 0.0750	192
	MUSE-RNN	0.2373 ± 0.0490	**190**

Table 6
Performance comparison on real-world HFT problem.

Algorithm	$\widehat{x}_{k+8,2} = f(\boldsymbol{x}_k)$		$\widehat{x}_{k+24,2} = f(\boldsymbol{x}_k)$	
	NDEI	#(Rules)	*NDEI*	#(Rules)
SAFLE	**0.131**	83	**0.192**	83
SAFL	0.170	7	0.244	7
DENFIS	1.598	12	1.582	12
eTS	0.183	6	0.271	7
SAFIS	0.554	20	0.779	14
RMCEFS	0.153	**4**	0.228	**6**

then-train evaluation method. The prediction performances of SAFLE on the two real-world problems (in terms of *NDEI*, #(Rules) and/or t_{exe}) are summarized in Tables 6 and 7, respectively. The results of the state-of-the-art approaches from the literature [18–20] are also presented in the two tables for benchmark comparison, and the best results are in bold.

Table 7
Performance comparison on S&P500 problem.

Algorithm	*NDEI*	#(Rules)	t_{exe}
SAFLE	0.0132	146	17.0
SAFL	**0.0121**	19	**1.8**
eTS	0.04	14	–
PANFIS	0.09	4	55.3
GENEFIS	0.07	**2**	48.3
SEFS	0.0182	**2**	2.3
EFS-SLAT	0.0156	23	–

It can be seen from Tables 5-7 that SAFLE outperforms all the comparative approaches on the seven non-stationary data stream prediction problems in terms of prediction accuracy (classification error and *NDEI*), and is only outperformed by SAFL on the S&P500 problem. This demonstrates the very strong capability of SAFLE to self-adapt to the rapidly changing patterns and mine useful information from the non-stationary data streams.

6. Conclusion

In this paper, a novel evolving fuzzy ensemble model, named SAFLE, is introduced for data stream prediction. The proposed SAFLE employs a group of SAFL-VSRP systems as its base models to self-learn and self-evolve from data streams on a chunk-by-chunk basis whilst simultaneously performing dimensionality compression, which effectively increases the diversity between base models and reduces their computational complexity. A dynamic pruning scheme is introduced to SAFLE, enabling it to autonomously identify and remove these error-prone components from its structure during the chunk-wise data stream learning process. To compensate the possible loss of key information caused by dimensionality compression, an ensemble inferencing scheme is further proposed to help SAFLE select only these elite base models for joint decision-making, thereby improving its prediction accuracy. Numerical examples based on a number of popular benchmark classification and regression problems demonstrate the superiority of the proposed SAFLE system.

There are a few considerations for future works.

Firstly, the use of dimensionality compression technique greatly improves the computational efficiency of SAFL-VSRP on high-dimensional problems, but the training time consumption is still higher than many alternative classification and regression approaches. This is mostly caused by the online updating of covariance matrices, though it is a standard practice of recursive least square algorithms used by first-order EFSs. To further improve the computational efficiency, one may need to employ computationally leaner algorithms for consequent parameter learning and updating.

Secondly, although SAFLE has a flexible structure and has the capability to remove less accurate base models, the number of ensemble components within the system is pre-fixed at this moment. It would be really helpful if a smarter approach is developed for SAFLE to self-determine the best number of base models based on the ensemble properties of data.

Thirdly, despite that SAFLE is able to achieve greater performance than a single SAFL system, it is not clear to which degree the stability of SAFLE is affected by the proposed random data splitting and base model pruning schemes. The stability issue needs to be investigated in future work.

Last but not the least, whether or not can VSRP or similar alternative techniques help zero-order EFS-based ensemble systems to achieve better performance remains a question to be answered, considering that zero-order EFSs are more computationally efficient compared with first-order ones.

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.

Data availability

Data will be made available on request.

Appendix

Sensitivity analysis

Further to the systematic evaluation presented in Section 5, the influences of externally controlled parameters, 1) the number of base models G; 2) the compression ratio, ε; 3) the chunk size, L; and 4) the split ratio, γ on the performance of the proposed SAFLE system are investigated. However, it needs to be stressed that the four externally controlled parameters are not problem-specific and can be determined without prior knowledge of the problem.

Table A1
Key information of regression problems for sensitivity analysis.

Dataset	Abbr.	#(Attributes)	#(Training Samples)	#(Testing Samples)
Mackey-Glass	MG	4	3000	500
California Housing	CH	8	10,320	10,320
Delta Ailerons	DA	5	3000	4129

Table A2
Influence of parameter settings.

Dataset	G	5	10	15	20	ε	0.25	0.5	0.75	1.0
MG	*NDEI*	0.0369	0.0343	0.0332	0.0328	*NDEI*	0.0653	0.0462	0.0343	0.0274
	t_{exe}	0.60	1.24	1.76	2.32	t_{exe}	1.09	1.18	1.24	1.27
CH	*NDEI*	0.0726	0.0723	0.0720	0.0717	*NDEI*	0.0893	0.0805	0.0723	0.0678
	t_{exe}	3.29	6.38	9.57	12.74	t_{exe}	5.91	5.99	6.38	6.67
DA	*NDEI*	0.0507	0.0505	0.0504	0.0503	*NDEI*	0.0543	0.0521	0.0505	0.0488
	t_{exe}	1.13	2.23	3.40	4.45	t_{exe}	2.09	2.21	2.23	2.34
Dataset	L	500	1000	2000	3000	γ	0.6	0.7	0.8	0.9
MG	*NDEI*	0.0315	0.0330	0.0343	0.0364	*NDEI*	0.0346	0.0338	0.0343	0.0350
	t_{exe}	1.22	1.16	1.16	1.02	t_{exe}	1.06	1.14	1.24	1.27
CH	*NDEI*	0.0725	0.0722	0.0723	0.0721	*NDEI*	0.0719	0.0725	0.0723	0.0722
	t_{exe}	6.11	6.26	6.38	6.42	t_{exe}	5.91	6.23	6.38	6.88
DA	*NDEI*	0.0504	0.0504	0.0505	0.0505	*NDEI*	0.0505	0.0504	0.0505	0.0505
	t_{exe}	2.18	2.20	2.23	1.97	t_{exe}	1.93	2.10	2.23	2.41

In this section, one classical benchmark regression problem, namely, Mackey-Glass time series prediction, and two real-world ones[7], 1) California Housing and 2) Delta Ailerons are employed for experiments. Key details of the three datasets are summarized by Table A1.

To start with, the influence of G on the performance of SAFLE is investigated. In this example, the value of G varies from 5, 10, 15 and 20. Other parameters are fixed as $\varepsilon = 0.75$, $L = 2000$ and $\gamma = 0.8$. The classification performance of SAFLE in terms of *NDEI* and t_{exe} are reported in Table A2.

Next, the influences of ε, L and γ on the system performance are investigated under the same experimental protocol with $G = 10$. During the experiments, the externally controlled parameters L and γ are firstly fixed as $L = 2000$ and $\gamma = 0.8$ with ε changed from 0.25, 0.5, 0.75 and 1.0. Then, the experiments are repeated with fixed ε and γ ($\varepsilon = 0.75$ and $\gamma = 0.8$) and L changed from 500 to 3000. Lastly, the same experiments are conducted with ε and L being fixed ($\varepsilon = 0.75$ and $L = 2000$) and γ varied from 0.6 to 0.9. Numerical results obtained with the three different experiment settings are tabulated in Tables A2 as well.

It can be seen from Table A2 that, firstly, by having more base models, namely, a greater G, SAFLE can achieve better prediction performance in terms of *NDEI*. However, this also means that SAFLE requires more computational resources. Secondly, a smaller compression ratio, ε can largely increase the computational efficiency by reducing the dimensionality of the covariance matrices associated with the fuzzy rules, but this may also lead to a significant loss of information, leading to worse prediction accuracy. Thirdly, the chunk size,

L has marginal influence on the system performance (both prediction accuracy and computational efficiency). This is due to the fact that the base models learn from streaming data sample-by-sample. On the other hand, one may note that SAFLE may struggle to identify these error-prone base models if L is too small because the amount of data samples stored in the validation data pools would be insufficient to represent the underlying data patterns. Nevertheless, this issue can be resolved by simply increasing the size of validation data pools. Lastly, the split ratio, γ will influence the computational efficiency since it determines the amounts of samples to be used for training and validation in each data chunk. A greater γ means more samples are used for training the base models, and vice versa.

Ablation analysis

In this section, ablation analysis is conducted to investigate the influence of dimensionality compression on the performances of SAFLE and its base models. In this example, four nonstationary synthetic datasets generated by HyperplaneGenerator API [8]with different dimensionalities varying from 8 to 64 are used. The four datasets are denoted as: HP_8, HP_{16}, HP_{32} and HP_{64}. The sizes of the four synthetic datasets are set to be 4000 (3000 for training and 1000 for testing) uniformly to facilitate computation. The classification performances of SAFLE and SAFL with and without VSRP on the four datasets are presented in Table A3 in terms of *Err* and t_{exe}.

From Table A3 one can see that VSRP effectively reduces the computational complexity of SAFLE and SAFL systems, especially in

[7] Available at: https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html.
[8] Available at: https://scikit-multiflow.readthedocs.io/en/stable/index.html.

Table A3
Results of ablation analysis.

Dataset	Meas.	SAFL		Δ	SAFLE		Δ
		VSRP	No VSRP		VSRP	No VSRP	
HP_8	*Err*	0.0352	0.0222	−0.0130 (-36.93%)	0.0260	0.0200	−0.0060 (–23.08%)
	t_{exe}	0.92	0.94	+0.02 (+2.17%)	8.22	8.36	+0.14 (+1.70%)
HP_{16}	*Err*	0.0305	0.0281	−0.0024 (-7.87%)	0.0293	0.0277	−0.0016 (-5.46%)
	t_{exe}	15.67	16.16	+0.49 (+3.13%)	135.40	138.40	+3.00 (+2.22%)
HP_{32}	*Err*	0.0353	0.0325	−0.0028 (-7.93%)	0.0313	0.0299	−0.0014 (-4.47%)
	t_{exe}	49.54	53.56	+4.02 (+8.11%)	376.55	409.66	+33.11 (+8.79%)
HP_{64}	*Err*	0.0522	0.0486	−0.0036 (-6.90%)	0.0461	0.0425	−0.0036 (-7.81%)
	t_{exe}	84.95	116.15	+31.20 (+36.73%)	619.69	832.67	+212.98 (-34.37%)

the cases where the data dimensionality is high. The higher the data dimensionality is, the greater improvement can be observed from the results in terms of computational efficiency. This demonstrates the effectiveness and validity of the proposed concept and general principles. On the other hand, one can see that both SAFLE and SAFL performs better in terms of classification accuracy without dimensionality compression because all the information from the data streams is preserved. Nevertheless, the decrease of prediction accuracy is within a reasonable range and, in fact, one can improve the prediction accuracy by increasing the number of ensemble components or increasing the compression ratio.

References

[1] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: a survey, Inf. Fusion 37 (2017) 132–156.
[2] D.H. Wolpert, M. Field, O. Error, The supervised learning no-free-lunch theorems, Soft Comput. Ind. (2002) 25–42.
[3] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (1–2) (2010) 1–39.
[4] L. Breiman, Bagging predictors, Mach. Learn. 24 (421) (1996) 123–140.
[5] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[6] A.M. Mohammed, E. Onieva, M. Woźniak, G. Martínez-Muñoz, An analysis of heuristic metrics for classifier ensemble pruning based on ordered aggregation, Pattern Recognit. 124 (2021), 108493.
[7] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, 2000.
[8] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in; *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
[9] P. Cunningham, S.J. Delany, K-nearest neighbour classifiers, Mult. Classif. Syst. 34 (2007) 1–17.
[10] Z. Zhou, J. Feng, Deep forest, Natl. Sci. Rev. 6 (1) (2019) 74–86.
[11] N.K. Kasabov, Q. Song, DENFIS : dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. Fuzzy Syst. 10 (2) (2002) 144–154.
[12] P.P. Angelov, D.P. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, IEEE Trans. Syst. Man, Cybern. - Part B Cybern. 34 (1) (2004) 484–498.
[13] P. Angelov, X. Zhou, Evolving fuzzy-rule based classifiers from data streams, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1462–1474.
[14] X. Gu, Q. Shen, A self-adaptive fuzzy learning system for streaming data prediction, Inf. Sci. (Ny) 579 (2021) 623–647.
[15] H.J. Rong, N. Sundararajan, G. Bin Huang, P. Saratchandran, Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction, Fuzzy Sets Syst. 157 (9) (2006) 1260–1275.
[16] E.D. Lughofer, FLEXFIS: a robust incremental learning approach for evolving Takagi-Sugeno fuzzy models, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1393–1410.
[17] M.M. Ferdaus, M. Pratama, S.G. Anavatti, M.A. Garratt, PALM: an incremental construction of hyperplanes for data stream regression, IEEE Trans. Fuzzy Syst. 27 (11) (2019) 2115–2129.
[18] D. Ge, X.J. Zeng, A self-evolving fuzzy system which learns dynamic threshold parameter by itself, IEEE Trans. Fuzzy Syst. 27 (8) (2018) 1625–1637.
[19] H. Rong, Z. Yang, P.K. Wong, Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system, IEEE Trans. Fuzzy Syst. 28 (9) (2019) 2277–2284.
[20] D. Ge, X.J. Zeng, Learning data streams online - an evolving fuzzy system approach with self-learning/adaptive thresholds, Inf. Sci. (Ny) 507 (2020) 172–184.
[21] R. Scherer, Designing boosting ensemble of relational fuzzy systems, Int. J. Neural Syst. 20 (5) (2010) 381–388.
[22] R. Scherer, An ensemble of logical-type neuro-fuzzy systems, Expert Syst. Appl. 38 (10) (2011) 13115–13120.
[23] B. Soua, A. Borgi, M. Tagina, An ensemble method for fuzzy rule-based classification systems, Knowl. Inf. Syst. 36 (2) (2013) 385–410.
[24] J. A. Iglesias, A. Ledezma, A. Sanchis, Ensemble method based on individual evolving classifiers, in: *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2013, pp. 56–61.
[25] M. Pratama, W. Pedrycz, E. Lughofer, Evolving ensemble fuzzy classifier, IEEE Trans. Fuzzy Syst. 26 (5) (2018) 2552–2567.
[26] M. Pratama, W. Pedrycz, G.I. Webb, An incremental construction of deep neuro fuzzy system for continual learning of nonstationary data streams, IEEE Trans. Fuzzy Syst. 28 (7) (2020) 1315–1328.
[27] X. Gu, Multilayer ensemble evolving fuzzy inference system, IEEE Trans. Fuzzy Syst. 29 (8) (2021) 2425–2431.
[28] X. Gu, P. Angelov, Multi-class fuzzily weighted adaptive boosting-based self-organising fuzzy inference ensemble systems for classification, IEEE Trans. Fuzzy Syst. 30 (9) (2022) 3722–3735.
[29] E. Lughofer, M. Pratama, I. Škrjanc, Online bagging of evolving fuzzy systems, Inf. Sci. (Ny) 570 (2021) 16–33.
[30] E. Lughofer, M. Pratama, Online sequential ensembling of predictive fuzzy systems, Evol. Syst. 13 (2) (2022) 361–386.
[31] E. Alves, R. Tanscheit, M. Vellasco, SENFIS - selected ensemble of fuzzy inference systems, IEEE Int. Conf. Fuzzy Syst. 2019-June (2019) 1–6.
[32] X. Gu, P. Angelov, Z. Zhao, Self-organizing fuzzy inference ensemble system for big streaming data classification, Knowledge-Based Syst. 218 (2021), 106870.
[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. Huang, H. Rong, Z. Yang, C. Vong, Jointly evolving and compressing fuzzy system for feature reduction and classification, Inf. Sci. (Ny) 579 (2021) 218–230.
[35] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., Wiley-Interscience, Chichester, West Sussex, UK, 2000.
[36] 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.

[37] Z. Yang, H. Rong, P.P. Angelov, Z. Yang, Statistically evolving fuzzy inference system for non-Gaussian noises, IEEE Trans. Fuzzy Syst. 30 (4) (2022) 2649–2664.
[38] U. Erkan, A precise and stable machine learning algorithm: eigenvalue classification (EigenClass), Neural Comput. Appl. 33 (10) (2021) 5381–5392.
[39] R.N. Patro, S. Subudhi, P.K. Biswal, F. Dell'Acqua, Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data, Int. J. Remote Sens. 40 (13) (2019) 4996–5024.
[40] G. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man, Cybern. Part B Cybern. 42 (2) (2012) 513–529.
[41] D.F. Specht, Probabilistic neural networks, Neural Networks 3 (1) (1990) 109–118.
[42] L. Breiman, Random forests, Mach. Learn. Proc. 45 (1) (2001) 5–32.
[43] J. Zhu, H. Zou, S. Rosset, T. Hastie, Multi-class AdaBoost, Stat. Interface 2 (3) (2009) 349–360.
[44] A. A. Rusu *et al.*, Progressive neural networks, *arXiv Prepr. arXiv1606.04671*, 2016, [Online]. Available: http://arxiv.org/abs/1606.04671.
[45] J. Yoon, E. Yang, J. Lee, S. J. Hwang, Lifelong learning with dynamically expandable networks, *arXiv Prepr. arXiv1708.01547*, 2017.
[46] Y.H. Jung, J. Goetz, A. Tewari, Online multiclass boosting, in: *Advances in Neural Information Processing Systems*, 2017, pp. 920–929.
[47] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: *International Conference on Machine Learning*, 2018, pp. 4548–4557.
[48] A. Ashfahani, M. Pratama, Autonomous deep learning: continual learning approach for dynamic environments, in: *SIAM International Conference on Data Mining*, 2019, pp. 666–674.
[49] 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.
[50] 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.