Self-Organizing Fuzzy Inference Ensemble System for Big Streaming Data Classification

Xiaowei Gu[a,*], Plamen Angelov[b], Zhijin Zhao[c]

[a] *Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK*
[b] *School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK*
[c] *School of Communication Engineering, Hangzhou Dianzi University, Hangzhou, 310018, China*

Abstract

An evolving intelligent system (EIS) is able to self-update its system structure and meta-parameters from streaming data. However, since the majority of EISs are implemented on a single-model architecture, their performances on large-scale, complex data streams are often limited. To address this deficiency, a novel self-organizing fuzzy inference ensemble framework is proposed in this paper. As the base learner of the proposed ensemble system, the self-organizing fuzzy inference system is capable of self-learning a highly transparent predictive model from streaming data on a chunk-by-chunk basis through a human-interpretable process. Very importantly, the base learner can continuously self-adjust its decision boundaries based on the inter-class and intra-class distances between prototypes identified from successive data chunks for higher classification precision. Thanks to its parallel distributed computing architecture, the proposed ensemble framework can achieve great classification precision while maintain high computational efficiency on large-scale problems. Numerical examples based on popular benchmark big data problems demonstrate the superior performance of the proposed approach over the state-of-the-art alternatives in terms of both classification accuracy and computational efficiency.

[*] Corresponding author
Email addresses: xig4@aber.ac.uk (Xiaowei Gu), p.angelov@lancaster.ac.uk (Plamen Angelov), zhaozj03@hdu.edu.cn (Zhijin Zhao)

Keywords: ensemble system, evolving intelligent system, large-scale data stream, prototypes.

1. Introduction

Evolving intelligent systems (EISs) are powerful learning machines for data stream processing with the strong capability to effectively approximate dynamically changing environments [1]. The majority of EISs are designed to learn from streaming data through a "one pass" procedure. They self-organize and self-develop the system structure and meta-parameters from data through a single scan, and can rapidly self-adjust to follow the drifts and/or shifts in data patterns [2]. Thus, the computational efficiency of EISs is usually higher than classical machine learning models, such as learning vector quantization [3], support vector machine (SVM) [4] and random forest (RF) [5], as well as the state-of-the-art deep learning-based approaches [6]. Thanks to the attractive features, EISs have been applied in many real-world applications successfully for handling streaming data and is now one of the most intensively researched areas in computational intelligence [7, 8].

Despite of the success they have achieved in time-critical applications [9], most of the existing EISs are built on a single-model architecture and they typically learn from streaming data on a sample-by-sample basis, resulting in some critical deficiencies when being applied to large-scale, high-dimensional, complex problems [10, 11]. It is often observed that the system structure of EISs can self-develop to an unfavorably large scale in such application scenarios, and their computational efficiency and model transparency are very low because of system obesity. In addition, since EISs typically learn from data on a sample-by-sample basis without revisiting historical data, they only have a partial understanding of data patterns. This also causes EISs forgetting the learned knowledge from previous data [1]. Thus, performances of single-model EISs on large-scale data streams are usually very limited.

To overcome these deficiencies, one possible solution is to construct an ensemble system composed of EISs to learn from streaming data on a chunk-by-chunk basis. It is well understood that ensemble learning is an efficient way to improve predictive accuracy of single-model base learners by decomposing a large-scale, complex problem into a number of small-scale, simpler sub-problems [12]. Moreover, learning from data chunks instead of individual data samples allows base learners to interpret local patterns of data streams better and still aligns to the concept of "one pass" learning without revisiting previous data chunks and iterative computation.

Therefore, in this paper, a novel ensemble learning approach named self-organizing fuzzy ensemble inference system (SOFEnsemble) is presented by putting the aforementioned idea into practice. The proposed SOFEnsemble employs the simplified self-organizing fuzzy inference system (SOFIS+) as its base learner. Unlike the original self-organizing fuzzy inference system (SOFIS) proposed in [13], the learning procedure of SOFIS+ is largely simplified, and the learning system now processes streaming data on a chunk-by-chunk basis. It identifies the more representative samples as prototypes from each data chunk and fuses the newly identified prototypes from the current data chunk with the previously identified ones from historical chunks based on the inter-class and intra-class distances in-between, resulting in more precise boundaries for decision-making. Furthermore, in the proposed ensemble architecture, training samples are randomly distributed to different base learners to improve the diversity, leading to better overall predictive accuracy. Meanwhile, this also reduces computational burden of each base learner, effectively improving the computational efficiency of the overall ensemble system. Numerical experiments on a variety of popular large-scale benchmark problems demonstrate the effectiveness and validity of the proposed concept, showing the promise of this work.

To summarize, key features setting this work apart from previous ones include: (1) a zero-order evolving fuzzy system self-learning from streaming data on a chunk-by-chunk basis; (2) a prototype-based approach to self-calibrate more precise decision boundaries from streaming data for classification; (3) a parallel ensemble architecture designed for large-scale data stream processing.

The remainder of this paper is organized as follows. A review of related works is provided in Section 2. Technical details of SOFIS are briefly recalled in Section 3 as the theoretical background. Section 4 presents the general architecture, learning and validation processes of SOFIS+. The ensemble architecture of SOFEnsemble is described in Section 5. Numerical examples are given in Section 6 as a proof of concept. This paper is concluded by Section 7.

2. Related Works

The concept of EISs was firstly introduced around the beginning of this century [9, 14, 15]. EISs are generally implemented in the forms of self-organizing, self-developing rule-based models [1, 9, 16] or neuro-fuzzy models [13, 14, 15, 17]. By offering highly transparent system structure and attractive prediction performance, EISs is currently a hotly studied area and are gaining increasing popularity over time-critical applications. Till now, many successful EISs have been developed, which include but are not limited to DENFIS [14], eTS [9], SAFIS [18], eClass [19], FLEXFIS [20], SOFMLS [21], GPFNN [17], PANFIS[22], GENEFIS[23], McIT2FIS[24], eT2Class [25], CNFS [26], ALMMo [16], LEOA [27], RMCEFS [28], SEFS [29] and PALM [30]. One may refer to [7, 8] for a more comprehensive review of recent works on EISs.

As single-model EISs often fail to produce reliable results for large-scale, complex problems, there have been some works in the literature that create an ensemble of EISs for better predictive performance on streaming data. The very first work combining ensemble learning with EISs was published in [31], where an ensemble system based on individual eClass0 [19] fuzzy classifiers was constructed to pursue better classification accuracy on streaming data. In [32], massively parallel deep fuzzy rule-based (DRB) ensemble classifiers were proposed for image classification as an alternative to deep learning-based approaches, demonstrating very promising performance. An ensemble based on three different types of EISs was created for weather time series prediction in [33]. However, as the individual base EISs involved in these works [31, 32] perform online learning from data streams on a sample-by-sample basis, their computational efficiency is inevitably low on large-scale problems especially when the dimensionality of data is very high. pENsemble proposed in [11] adopts a novel dynamic ensemble structure to tackle the concept drifts in data streams by evolving in both ensemble level and base learner level. pENsemble automatically initializes new base learners and prunes stale ones to follow the changing data patterns. Nonetheless, the fully evolving ensemble structure of pENsemble causes quick forgetting of previously mined knowledge from historical data, making the ensemble model less suitable for large-scale classification problems.

Till now, there are only a few fuzzy classifiers proposed for managing large-scale data. The earliest one is the so-called Chi-FRBCS-BigData proposed in [34, 35], which is a fuzzy rule-based classification system (FRBCS) that uses the MapReduce programming model [36] to learn and fuse fuzzy rule bases from big data using Chi et al.'s approach [37]. Chi-FRBCS-BigData firstly divides the input dataset into several chunks and distributes them to different computational units to generate fuzzy rules individually. Then, all the fuzzy rule bases are fused together for classifying unlabeled data. An improved version named Chi-FRBCS-BigDataCS was further proposed in [38] to tackle imbalanced large-scale classification problems by involving cost-sensitive learning. Nonetheless, since the weights of fuzzy rules learned from each data chunk is subject to the proportion and distribution of the specific subset of input data, the performances of both Chi-FRBCS-BigData and Chi-FRBCS-BigDataCS are highly influenced by the proportion and distribution of the classes in the individual data chunks. The more chunks the original input dataset is split to, the worse both algorithms perform because the number of samples in each chunk becomes smaller. To overcome this issue, a new version of Chi-FRBCS-BigData was proposed in [39] by firstly learning a preliminary fuzzy rule base from different data chunks and calculating rule weights using all the input data afterwards. Moreover, a distributed fuzzy decision tree (DFDT) learning method was also proposed in [40] upon the MapReduce scheme.

3. Preliminaries

In this section, technical details of SOFIS introduced in [13] are briefly recalled to make this paper self-contained.

First of all, let $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n, ..., \boldsymbol{x}_N\}$ be a particular data set/stream in a M dimensional data space, $\Re^M$. The subscript n denotes the time instance at which the n^{th} sample, $\boldsymbol{x}_n$ is observed. It is assumed that $\{\boldsymbol{x}\}$ is composed of samples C different classes. Therefore, $\{\boldsymbol{x}\}$ can be divided into C subsets according to the class labels, namely, $\{\boldsymbol{x}\}^c = \{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, ..., \boldsymbol{x}_{N^c}^c\} \subset \{\boldsymbol{x}\}$ ($c = 1, 2, ..., C$), where N^c is the cardinality of $\{\boldsymbol{x}\}^c$; and there is $N = \sum_{c=1}^{C} N^c$. It is also assumed that different data samples of the same classes may have exactly the same values, namely, $\boldsymbol{x}_i^c = \boldsymbol{x}_j^c$ for $i \neq j$. Thus, the set of unique data samples of the c^{th} class is denoted as $\{\boldsymbol{u}\}^c = \{\boldsymbol{u}_1^c, \boldsymbol{u}_2^c, ..., \boldsymbol{u}_{Q^c}^c\} \subseteq \{\boldsymbol{x}\}^c$ with the corresponding occurrence frequencies denoted as $\{f\}^c = \{f_1^c, f_2^c, ..., f_{Q^c}^c\}$, where Q^c is the cardinality of $\{\boldsymbol{u}\}^c$; f_i^c is the occurrence frequency of $\boldsymbol{u}_i^c$; and there is $N^c = \sum_{i=1}^{Q^c} f_i^c$. Without loss of generality, Euclidean distance is used by default.

3.1. Architecture

The architecture of SOFIS is illustrated in Fig. 1. It can be observed from Fig. 1 that the system is composed of C data processors for prototype identification and a fuzzy rule base, which is composed of C massively parallel fuzzy rules formed by prototypes (namely, the most representative samples) as follows [13].

$$\begin{aligned} \boldsymbol{R}^c :\ & IF\ (\boldsymbol{x} \sim \boldsymbol{p}_1^c)\ OR\ (\boldsymbol{x} \sim \boldsymbol{p}_2^c)\ OR\ ...\ OR\ (\boldsymbol{x} \sim \boldsymbol{p}_{P^c}^c) \\ & THEN\ (Class\ c) \end{aligned} \tag{1}$$

where $\boldsymbol{p}_i^c$ is the i^{th} ($i = 1, 2, ..., P^c$) prototype of $\boldsymbol{R}^c$; P^c is the total number of prototypes identified from data. These prototypes are connected by logical "OR" connectives, and thus, $\boldsymbol{R}^c$ can be viewed as a parallel ensemble of multiple simpler fuzzy rules in the following form [19].

$$\boldsymbol{R}_i^c: \; IF\ (\boldsymbol{x} \sim \boldsymbol{p}_i^c)\ \ THEN\ (Class\ c) \tag{2}$$

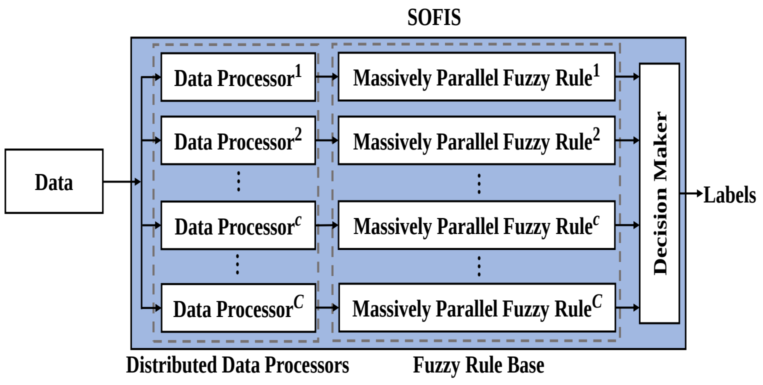

SOFIS
Data
Data Processor^1
Data Processor^2
Data Processor^c
Data Processor^C
Massively Parallel Fuzzy Rule^1
Massively Parallel Fuzzy Rule^2
Massively Parallel Fuzzy Rule^c
Massively Parallel Fuzzy Rule^C
Decision Maker
Labels
Distributed Data Processors
Fuzzy Rule Base

Figure 1: Architecture of SOFIS.

During the learning stage, SOFIS is able to self-organize a set of massively parallel fuzzy rules from static data samples based on their ensemble properties and mutual distances. After being primed offline, SOFIS can continue to self-update its fuzzy rule base with newly arrived streaming data on sample-by-sample basis and self-expand its knowledge base in a recursive manner. During the validation stage, SOFIS predicts the class label of each unlabeled sample based on the set of scores of confidence produced by its fuzzy rules.

The offline learning, online updating and validation processes are summarized in the next three subsections, respectively. The level of granularity used by SOFIS is set as G (a positive integer).

3.2. Offline Learning Process

The offline learning process of SOFIS is described as follows [13].

Stage 1. Identifying local maxima from data

Given a static dataset, SOFIS firstly divides the input data, $\{\boldsymbol{x}\}$ into C subsets according to the class labels, denoted by $\{\boldsymbol{x}\}^1$, $\{\boldsymbol{x}\}^2$,...,$\{\boldsymbol{x}\}^C$. Then, each subset is passed to the corresponding data processor for prototype identification.

After the c^{th} $(c = 1, 2, ..., C)$ data processor receives $\{\boldsymbol{x}\}^c$, it firstly calculates the multimodal data density values at the observed unique samples using equation (3) $(i = 1, 2, ..., Q^c)$ [13].

$$D^{MM}(\boldsymbol{u}_i^c) = \frac{f_i^c}{1 + \frac{||\boldsymbol{u}_i^c - \boldsymbol{\mu}_{N^c}^c||^2}{X_{N^c}^c - ||\boldsymbol{\mu}_{N^c}^c||^2}} \tag{3}$$

where $\boldsymbol{\mu}_{N^c}^c$ and $X_{N^c}^c$ are the respective arithmetic means of $\{\boldsymbol{x}\}^c$ and $\{||\boldsymbol{x}||^2\}^c$.

The data processor then ranks the unique samples using equation (4) according to their multimodal density values and mutual distances. The ranked unique data sample set is re-denoted as $\{\boldsymbol{r}\}^c = \{\boldsymbol{r}_1^c, \boldsymbol{r}_2^c, ..., \boldsymbol{r}_{Q^c}^c\}$.

$$\begin{cases} \boldsymbol{r}_i^c = \arg\max_{\boldsymbol{u}\in\{\boldsymbol{u}\}^c}(D^{MM}(\boldsymbol{u})); & i = 1 \\ \boldsymbol{r}_i^c = \arg\min_{\boldsymbol{u}\in\{\boldsymbol{u}\}^c; \boldsymbol{u}\neq\boldsymbol{r}_1^c, \boldsymbol{r}_2^c, ..., \boldsymbol{r}_{i-1}^c}(||\boldsymbol{u} - \boldsymbol{r}_{i-1}^c||^2); & else \end{cases} \tag{4}$$

Local maxima of multimodal density, denoted as $\{\boldsymbol{l}\}^c$, are then identified by the data processor using Condition 1 [13].

$$\begin{aligned} \text{Condition 1: } & \textit{If } (D^{MM}(\boldsymbol{r}_i^c) > D^{MM}(\boldsymbol{r}_{i-1}^c)) \\ & \textit{And } (D^{MM}(\boldsymbol{r}_i^c) > D^{MM}(\boldsymbol{r}_{i+1}^c)) \\ & \textit{Then } (\{\boldsymbol{l}\}^c \leftarrow \{\boldsymbol{l}\}^c \cup \{\boldsymbol{r}_i^c\}) \end{aligned} \tag{5}$$

Then, Voronoi tessellations are formed around these local maxima by using them to attract nearby data samples of the same class creating micro-clusters $(i = 1, 2, ..., N^c)$:

$$\mathbb{C}_{j^*}^c \leftarrow \mathbb{C}_{j^*}^c \cup \{\boldsymbol{x}_i^c\}; \quad j^* = \underset{j=1,2,...,\hat{Q}^c}{\arg\min}(||\boldsymbol{x}_i^c - \boldsymbol{l}_j^c||^2) \tag{6}$$

where $\mathbb{C}_j^c$ is the micro-cluster formed around $\boldsymbol{l}_j^c$; $\hat{Q}^c$ is the cardinality of $\{\boldsymbol{l}\}^c$.

After this, the c^{th} data processor extracts the arithmetic means of the micro-clusters as raw prototypes, denoted as $\{\hat{\boldsymbol{l}}\}^c$, and enters the next processing stage.

Stage 2. Identifying prototypes from local maxima

Once the c^{th} $(c = 1, 2, ..., C)$ data processor enters stage 2, it firstly estimates the date-driven threshold γ_G^c using equation (7) based on the mutual distances between samples of $\{\boldsymbol{x}\}^c$ and the level of granularity controlled externally by users:

$$\gamma_g^c = \frac{1}{N_g^c} \sum_{\substack{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}^c;\boldsymbol{x}\neq\boldsymbol{y};\\ ||\boldsymbol{x}-\boldsymbol{y}||^2\leq\gamma_{g-1}^c}} ||\boldsymbol{x}-\boldsymbol{y}||^2 \tag{7}$$

where $g = 1, 2, ..., G$; $\gamma_0^c = 2(X_{N^c}^c - ||\boldsymbol{\mu}_{N^c}^c||^2)$, which is the average squared Euclidean distance between any two samples in $\{\boldsymbol{x}\}^c$; N_g^c is the number of sample pairs in $\{\boldsymbol{x}\}^c$ between which the distance is smaller than γ_{g-1}^c.

Note that γ_G^c provides an empirical estimation of the radius of area of influence around each prototype at the specific level of granularity, condensing the mutual distribution information mined from data. Importantly, γ_G^c is derived directly from data without prior knowledge of the problem and is guaranteed to be meaningful.

Then, the c^{th} data processor identifies a set of neighboring raw prototypes, $\{\hat{\boldsymbol{l}}\}_i^c$ for each raw prototype, $\hat{\boldsymbol{l}}_i^c$ using Condition 2 [13].

$$\begin{aligned} &\text{Condition 2: } \textit{If } (||\hat{\boldsymbol{l}}_i^c - \hat{\boldsymbol{l}}_j^c||^2 \leq \gamma_G^c) \\ &\qquad\qquad \textit{Then } (\{\hat{\boldsymbol{l}}\}_i^c \leftarrow \{\hat{\boldsymbol{l}}\}_i^c \cup \{\hat{\boldsymbol{l}}_j^c\}) \end{aligned} \tag{8}$$

where $i, j = 1, 2, ..., \hat{Q}^c$ and $i \neq j$.

The more representative raw prototypes, denoted as $\{\hat{\boldsymbol{p}}\}^c$, are identified from $\{\hat{\boldsymbol{l}}\}^c$ using Condition 3 ($i = 1, 2, ..., \hat{Q}^c$) [13].

$$\begin{aligned} &\text{Condition 3: } \textit{If } (D^{MM}(\hat{\boldsymbol{l}}_i^c) > \max_{\boldsymbol{l}\in\{\hat{\boldsymbol{l}}\}_i^c} (D^{MM}(\boldsymbol{l}))) \\ &\qquad\qquad \textit{Then } (\{\hat{\boldsymbol{p}}\}^c \leftarrow \{\hat{\boldsymbol{p}}\}^c \cup \{\hat{\boldsymbol{l}}_i^c\}) \end{aligned} \tag{9}$$

$\{\hat{\boldsymbol{p}}\}^c$ are used for forming Voronoi tessellations with $\{\boldsymbol{x}\}^c$ ($i = 1, 2, ..., N^c$):

$$\mathbf{C}_{j^*}^c \leftarrow \mathbf{C}_{j^*}^c \cup \{\boldsymbol{x}_i^c\}; \quad j^* = \underset{j=1,2,...,P^c}{\arg\min} (||\boldsymbol{x}_i^c - \hat{\boldsymbol{p}}_j^c||^2) \tag{10}$$

where $\mathbf{C}_j^c$ is the cluster formed around $\hat{\boldsymbol{p}}_j^c$; P^c is the cardinality of $\{\hat{\boldsymbol{p}}\}^c$.

Finally, the c^{th} data processor extracts the arithmetic means of these clusters, denoted as $\{\boldsymbol{p}\}^c$ and constructs $\boldsymbol{R}^c$ in the same form of equation (1). After all the data processors have finished the prototype extraction process and built the massively parallel fuzzy rules, the offline learning process is completed. However, it is worth noting that each data processor is working independently from others. In other words, there is no interaction between any two processors during prototype identification.

3.3. Online Updating Process

After being primed offline, SOFIS can continue to self-update its system structure and meta-parameters from streaming data on a sample-by-sample basis. The algorithmic procedure is summarized as follows [13].

Assuming that the newly observed sample belongs to the c^{th} class, denoted as $\boldsymbol{x}^c_{N^c+1}$, after the c^{th} data processor receives this new sample, it firstly updates the data-driven threshold using equation (11) [13]:

$$\gamma^c_G \leftarrow \frac{X^c_{N^c+1} - ||\boldsymbol{\mu}^c_{N^c+1}||^2}{X^c_{N^c} - ||\boldsymbol{\mu}^c_{N^c}||^2}\gamma^c_G \tag{11}$$

where $\boldsymbol{\mu}^c_{N^c+1}$ and $X^c_{N^c+1}$ are calculated using equation (12)

$$\boldsymbol{\mu}^c_{N^c+1} = \frac{N^c\boldsymbol{\mu}^c_{N^c} + \boldsymbol{x}^c_{N^c+1}}{N^c+1}; \quad X^c_{N^c+1} = \frac{N^cX^c_{N^c} + ||\boldsymbol{x}^c_{N^c+1}||^2}{N^c+1} \tag{12}$$

Then, the c^{th} data process determines whether $\boldsymbol{x}^c_{N^c+1}$ has the potential to become a new prototype using Condition 4 [13].

$$\begin{aligned} \text{Condition 4: } & \textit{If } (D(\boldsymbol{x}^c_{N^c+1}) > \max_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}(D(\boldsymbol{p}))) \\ & \textit{Or } (D(\boldsymbol{x}^c_{N^c+1}) < \min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}(D(\boldsymbol{p}))) \\ & \textit{Or } (\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}(||\boldsymbol{p} - \boldsymbol{x}^c_{N^c+1}||^2) > \gamma^c_G) \\ & \textit{Then } (\boldsymbol{x}^c_{N^c+1} \textit{ becomes a new prototype}) \end{aligned} \tag{13}$$

where $D(\boldsymbol{z})$ is the data density value at $\boldsymbol{z}$ calculated using equation (14); $\boldsymbol{z} = \boldsymbol{x}^c_{N^c+1}, \boldsymbol{p}^c_1, \boldsymbol{p}^c_2, ..., \boldsymbol{p}^c_{P_c}$.

$$D(\boldsymbol{z}) = \frac{1}{1 + \frac{||\boldsymbol{z}-\boldsymbol{\mu}^c_{N^c+1}||^2}{X^c_{N^c+1} - ||\boldsymbol{\mu}^c_{N^c+1}||^2}} \tag{14}$$

If Condition 4 is satisfied, $\boldsymbol{x}^c_{N^c+1}$ is recognized as a new prototype:

$$P^c \leftarrow P^c + 1; \quad \boldsymbol{p}^c_{P^c} \leftarrow \boldsymbol{x}^c_{N^c+1}; \quad \mathbf{C}^c_{P^c} \leftarrow \{\boldsymbol{x}^c_{N^c+1}\}; \quad S^c_{P^c} \leftarrow 1 \tag{15}$$

Otherwise, $\boldsymbol{x}^c_{N^c+1}$ is used for updating the meta-parameters of the nearest prototype:

$$\boldsymbol{p}^c_{n^*} \leftarrow \frac{S^c_{n^*}\boldsymbol{p}^c_{n^*} + \boldsymbol{x}^c_{N^c+1}}{S^c_{n^*}+1}; \quad \mathbf{C}^c_{n^*} \leftarrow \mathbf{C}^c_{n^*} \cup \{\boldsymbol{x}^c_{N^c+1}\}; \quad S^c_{n^*} \leftarrow S^c_{n^*}+1 \tag{16}$$

where $n^* = \arg\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}(||\boldsymbol{x}^c_{N^c+1} - \boldsymbol{p}||^2)$.

After this, the c^{th} data process updates $\boldsymbol{R}^c$ with the latest prototypes, $\{\boldsymbol{p}\}^c$ and enters the a new processing cycle for the next data sample ($N^c \leftarrow N^c + 1$).

3.4. Validation Process

During validation, every massively parallel fuzzy rule, $\boldsymbol{R}^c$ ($c = 1, 2, ..., C$) will produce a score of confidence on each unlabeled sample $\boldsymbol{x}$ calculated based on the similarity between $\boldsymbol{x}$ and its prototypes, $\mathbf{P}^c$ ($c = 1, 2, ..., C$) [13]:

$$\lambda^c(\boldsymbol{x}) = \max_{\boldsymbol{p}\in\{\boldsymbol{p}\}^c}(e^{-||\boldsymbol{x}-\boldsymbol{p}||^2}) \tag{17}$$

The class label of $\boldsymbol{x}$ is determined based on the C scores of confidence produced by the C fuzzy rules:

$$label(\boldsymbol{x}) \leftarrow class\ i^*; \quad i^* \leftarrow \underset{c=1,2,\ldots,C}{\arg\max}\,(\lambda^c(\boldsymbol{x})) \tag{18}$$

4. SOFIS+

As stated in Section 1, SOFIS+ is a simplified version of SOFIS [13] that learns from streaming data on a chunk-by-chunk basis. For each arriving data chunk, SOFIS+ firstly measures the ensemble properties of the observed samples to gain a better understanding of data distribution. Then, it learns from the data chunk on a sample-by-sample basis to identify the most representative samples as prototypes. These newly identified prototypes are further fused with existing prototypes within the knowledge base to build more precise decision-boundaries for classification. SOFIS+ will not revisit the processed samples and will discard the data chunk once the current learning cycle is completed to guarantee its computation- and memory- efficiency. Thus, it aligns closely to the "one pass" learning concept [11].

To summarize, SOFIS+ differs from SOFIS in the following key aspects:

1. it learns from data streams on a chunk-by-chunk basis, and its learning process is much simplified;
2. it considers both the inter-class and intra-class distances between the identified prototypes to construct more precise classification boundaries;
3. it uses prototypes to directly represent the underlying data patterns without using clusters.

4.1. Architecture

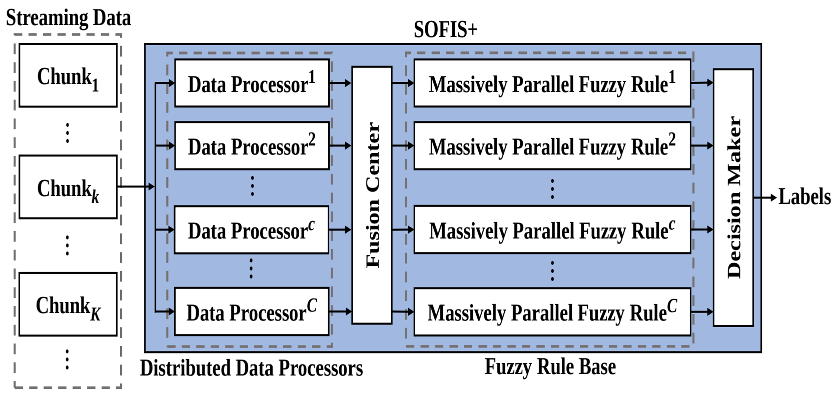

Streaming Data
SOFIS+
Chunk_1
Chunk_k
Chunk_K
Data Processor^1
Data Processor^2
Data Processor^c
Data Processor^C
Fusion Center
$\text{Massively Parallel Fuzzy Rule}^1$
$\text{Massively Parallel Fuzzy Rule}^2$
$\text{Massively Parallel Fuzzy Rule}^c$
$\text{Massively Parallel Fuzzy Rule}^C$
Decision Maker
Labels
Distributed Data Processors
Fuzzy Rule Base

Figure 2: Architecture of SOFIS+.

The diagram of the architecture of SOFIS+ is given in Fig. 2. It can be observed from Fig. 2 that the system is composed of C data processors for extracting prototypes from input data, a fusion center for fusing newly identified prototypes to the existing knowledge base, and a fuzzy rule base, which consists of C massively parallel fuzzy rules in the same form as equation (1).

During the learning stage, given a new data chunk $\{\boldsymbol{x}\}_k = \{\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, ..., \boldsymbol{x}_{k,L}\}$ ($k = 1, 2, ..., K$; L denotes the chunk size), SOFIS+ firstly divides it into C subsets based on class labels, namely, $\{\boldsymbol{x}\}_k^c = \{\boldsymbol{x}_{k,1}^c, \boldsymbol{x}_{k,2}^c, ..., \boldsymbol{x}_{k,L^c}^c\} \subset \{\boldsymbol{x}\}_k$, where $c = 1, 2, ..., C$; L^c is the cardinality of $\{\boldsymbol{x}\}_k^c$; and there is $L = \sum_{c=1}^{C} L_k^c$. Every data processor within SOFIS+ then grasps its corresponding subset for prototype identification. The newly identified prototypes are collected by the fusion center. The fusion center will then compare these new prototypes between each other as well as with prototypes identified from previously processed chunks to select out the best candidates to help the system build more precise decision boundaries for classification. After this, the fuzzy rule base is updated accordingly with these selected prototypes. During the validation stage, SOFIS+ follows the same decision-making procedure of SOFIS but with a modified expression for calculating the scores of confidence.

In the following two subsections, the learning and validation processes of SOFIS+ will be detailed.

4.2. Learning Process

The main algorithmic procedure of the chunk-by-chunk online learning process of SOFIS+ is as follows [13]. The level of granularity of SOFIS+ is set as G as well.

Stage 1. Identifying prototypes

When a new chunk, $\{\boldsymbol{x}\}_k$ is available, SOFIS+ firstly divides it into C subsets $\{\boldsymbol{x}\}_k^1, \{\boldsymbol{x}\}_k^2, ..., \{\boldsymbol{x}\}_k^C$ according to class labels and passes them to the corresponding data processors.

Once the c^{th} $(c = 1, 2, ..., C)$ data processor receives $\{\boldsymbol{x}\}_k^c$, it firstly estimates the date-driven threshold $\gamma_{k,G}^c$ using equation (7) based on the mutual distances between samples within $\{\boldsymbol{x}\}_k^c$ and the level of granularity controlled externally by users.

The first prototype $(P_k^c \leftarrow 1)$ of the c^{th} class identified at the current learning cycle is initialized as the first sample of $\{\boldsymbol{x}\}_k^c$, namely, $\boldsymbol{p}_{k,P_k^c}^c \leftarrow \boldsymbol{x}_{k,1}^c$. The corresponding support (number of associated data samples) of $\boldsymbol{p}_{k,P_k^c}^c$ is set as $S_{k,P_k^c}^c \leftarrow 1$. Then, the remaining prototypes are identified one-by-one using Condition 5 $(j = 2, 3, ..., L_k^c)$:

$$\begin{gathered}\text{Condition 5: } \textit{If } (\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}_k^c}(||\boldsymbol{x}_{k,j}^c - \boldsymbol{p}_{k,n^*}^c||^2) > \gamma_{k,G}^c) \\ \textit{Then } (\boldsymbol{x}_{k,j}^c \textit{ is a new prototype})\end{gathered} \tag{19}$$

If Condition 5 is satisfied, $\boldsymbol{x}_{k,j}^c$ is recognized as a new prototype:

$$P_k^c \leftarrow P_k^c + 1; \quad \boldsymbol{p}_{k,P_k^c}^c \leftarrow \boldsymbol{x}_{k,j}^c; \quad S_{k,P_k^c}^c \leftarrow 1 \tag{20}$$

Otherwise, $\boldsymbol{x}_{k,j}^c$ is used for updating the nearest prototype as follows [13]:

$$\boldsymbol{p}_{k,n^*}^c \leftarrow \frac{S_{k,n^*}^c \boldsymbol{p}_{k,n^*}^c + \boldsymbol{x}_{k,j}^c}{S_{k,n^*}^c + 1}; \quad S_{k,n^*}^c \leftarrow S_{k,n^*}^c + 1 \tag{21}$$

where $n^* = \arg\min_{\boldsymbol{p}\in\{\boldsymbol{p}\}_k^c}(||\boldsymbol{x}_{k,j}^c - \boldsymbol{p}_{k,n^*}^c||^2)$. The prototypes identified from $\{\boldsymbol{x}\}_k^c$ are denoted as $\{\boldsymbol{p}\}_k^c$.

Once the C data processors have completed the prototype identification processes, they will pass $\{\boldsymbol{p}\}_k^c$ $(c = 1, 2, ..., C)$ to the fusion center. Then, the learning algorithm enters the next stage.

Stage 2. Self-calibrating decision boundaries

After fusion center receives the prototypes, $\{\boldsymbol{p}\}_k^1, \{\boldsymbol{p}\}_k^2, ..., \{\boldsymbol{p}\}_k^C$ identified from the current data chunk, $\{\boldsymbol{x}\}_k$ from the C data processors, it will initialize the knowledge base, denoted by $\mathbf{P}^c$ and the fuzzy rule, denoted by $\boldsymbol{R}^c$ (in the form of equation (1)) of each class $(c = 1, 2, ..., C)$ with the corresponding prototypes, $\{\boldsymbol{p}\}_k^c$ if this is the first data chunk $(k = 1)$.

Otherwise, for each class (assuming the c^{th} one; $c = 1, 2, ..., C$), the fusion center firstly compares $\{\boldsymbol{p}\}_k^c$ with all existing prototypes of the same class identified from previous processing cycles, and selects out the candidate prototypes from $\{\boldsymbol{p}\}_k^c$ to update the knowledge base $\mathbf{P}^c$ by Condition 6.

$$\begin{aligned} &\text{Condition 6: } \textit{If } (\min_{\boldsymbol{q}\in\mathbf{P}^c}(||\boldsymbol{q} - \boldsymbol{p}_{k,j}^c||^2) > \bar{\gamma}_G^c) \\ &\qquad \textit{Then } (\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \{\boldsymbol{p}_{k,j}^c\}) \end{aligned} \tag{22}$$

where $\bar{\gamma}_G^c = \frac{1}{k}\sum_{i=1}^k \gamma_{i,G}^c$, which is the average radius of area of influence of prototypes calculated based on all historical samples of the c^{th} class. Condition 6 helps SOFIS+ to self-expand its knowledge base with only these new prototypes that are distinctive from the existing ones. This effectively reduces the system complexity because new prototypes that represent familiar data patterns recognized from previous learning cycles are avoided being added to the system.

Then, Condition 7 is used for identifying the new prototypes that are closer to prototypes of other classes identified at the current learning cycle and adding them to $\mathbf{P}^c$:

$$\begin{gathered}\text{Condition 7: } \textit{If } (\min_{\boldsymbol{q}\in\{\boldsymbol{p}\}_k^l}(||\boldsymbol{q}-\boldsymbol{p}_{k,j}^c||^2) \leq 2\bar{\gamma}_G^l \ \forall\ l \neq c) \\ \textit{Then } (\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \{\boldsymbol{p}_{k,j}^c\})\end{gathered} \tag{23}$$

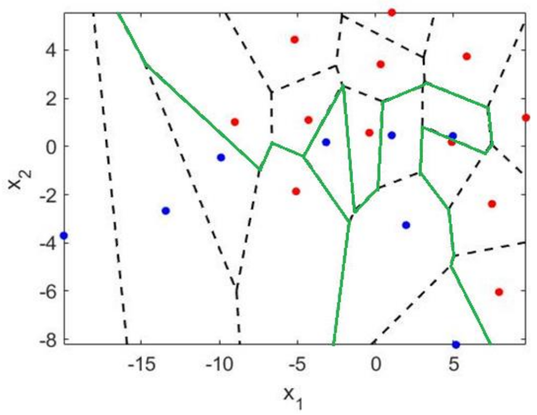
Condition 7 enables SOFIS+ to effectively self-improve the fineness of the learned decision boundaries from data for more precise classification. Then, the fuzzy rule, $\boldsymbol{R}^c$ is updated with the updated knowledge base $\mathbf{P}^c$. In the end, after all the knowledge bases and fuzzy rules of the C classes are updated, the learning algorithm goes back to stage 1 for processing the next available data chunk ($k \leftarrow k+1$).

The main algorithmic procedure for SOFIS+ identification is summarized in Algorithm 1. It is worth to be noticed that during each learning cycle, stage 1 is performed by the individual data processors, and stage 2 is performed by the fusion center. In addition, since the prototype identification processes from data samples of the C classes are performed separately per class, these data processors can be further implemented in parallel to speed up the learning process.

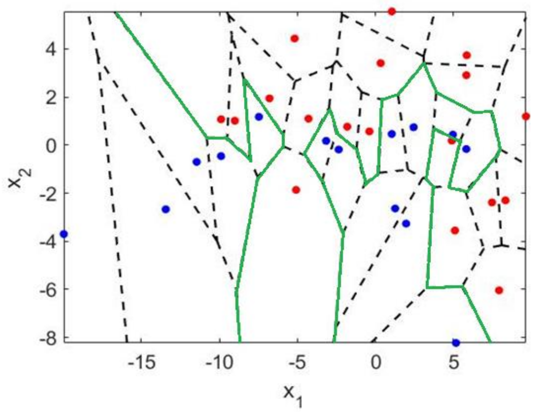
An illustrative example is given in Fig. 3, demonstrating how SOFIS+ learns finer decision boundaries from two successive chunks of streaming data, where the dots in two different colors ("red" and "blue") stand for prototypes identified from samples of two different classes; the black dash lines represent the boundaries formed by Voronoi tessellations; the green lines are the decision boundaries. One can observe from Fig. 3 that SOFIS+ is able to learn coarse decision boundaries from the first data chunk, and then, SOFIS+ further adjusts the decision boundaries based on the learning outcome from the second chunk.

4.3. Validation Process

During the validation stage, for each unlabeled sample $\boldsymbol{x}$, a score of confidence is produced by every fuzzy rule, $\boldsymbol{R}^c$ in terms of the maximum similarity



(a) Learned decision boundaries from chunk_1



(b) Updated decision boundaries from chunk_2

Figure 3: Illustration of learning more precise decision boundaries from streaming data.

Algorithm 1 SOFIS+ Identification.

```
while (a new chunk $\{\boldsymbol{x}\}_k^c$ is available) do
   //          Stage 1. Identifying prototypes          //
   for $c = 1$ to $C$ do
      derive $\gamma_{k,G}^c$ from $\{\boldsymbol{x}\}_k^c$ by (7);
      $P_k^c \leftarrow 1$;
      $\boldsymbol{p}_{k,P_k^c}^c \leftarrow \boldsymbol{x}_{k,1}^c$; $S_{k,P_k^c}^c \leftarrow 1$;
      for $j = 2$ to $L_k^c$ do
         if (Condition 5 is satisfied) then
            add $\boldsymbol{x}_{k,j}^c$ as a new prototype by (20);
         else
            update $\boldsymbol{p}_{k,n^*}^c$ with $\boldsymbol{x}_{k,j}^c$ by (21);
         end if
      end for
   end for
   //     Stage 2. Self-calibrating decision boundaries     //
   for $c = 1$ to $C$ do
      if ($k = 1$) then
         $\mathbf{P}^c \leftarrow \{\boldsymbol{p}\}_k^c$;
         initialize $\boldsymbol{R}^c$ with $\mathbf{P}^c$;
      else
         expand $\mathbf{P}^c$ with $\{\boldsymbol{p}\}_k^c$ by Conditions 6 and 7;
         update $\boldsymbol{R}^c$ with $\mathbf{P}^c$;
      end if
   end for
end while
```

between $\boldsymbol{x}$ and its prototypes, $\mathbf{P}^c$ $(c = 1, 2, ..., C)$:

$$\lambda^c(\boldsymbol{x}) = \max_{\boldsymbol{p}\in\mathbf{P}^c}(e^{-\frac{||\boldsymbol{x}-\boldsymbol{p}||^2}{X_N-||\boldsymbol{\mu}_N||^2}}) \tag{24}$$

where $\boldsymbol{\mu}_N$ and X_N are the respective arithmetic means of $\{\boldsymbol{x}\}$ and $\{||\boldsymbol{x}||^2\}$, both of them can be calculated recursively [13]. Note that $X_N - ||\boldsymbol{\mu}_N||^2$ is the half of the average squared Euclidean distance between any two observed samples. It serves as a normalization factor in equation (24), effectively increasing the differences between scores of confidence produced by different fuzzy rules.

The class label of $\boldsymbol{x}$ is determined based on the C scores of confidence produced by the C fuzzy rules using equation (18) following the "winner takes all" principle.

5. Proposed SOFEnsemble

In this section, technical details of the proposed SOFEnsemble are presented. Architecture of the proposed ensemble system is depicted in Fig. 4. As shown by this figure, the proposed SOFEnsemble is composed of one data distributor, F data pools, F base classifiers implemented in parallel (F is predefined in advance) and one decision fusion module. All base learners have the same SOFIS+ architecture as given by Fig. 2.

5.1. Learning Policy

During the learning process, the data distributor receives the streaming data arrived either in samples or in chunks, and randomly distributes the observed samples to F data pools. The F data pools have the uniform size of L, and they collected streaming samples for the respective base learners as the input data chunks. However, unlike other existing ensemble frameworks [11, 31, 32] where each sample can only be passed to one of the base learners, the data distributor randomly assigns each sample to H data pools (H is an integer; $1 < H < F$). The proposed data distribution policy guarantees the diversity of the F base learners, and helps each base learner to learn more precise decision boundaries from data, resulting in better classification performance. Once a data pool is full, all the samples in the pool are packed as a data chunk and passed to the corresponding base classifier. Then, the data pool becomes empty again and continues to collect new samples to build the next data chunk.

After the base learner receives a new data chunk, it begins a new learning cycle. The base learner firstly identifies new prototypes from the data chunk and then uses them to update the knowledge base following the algorithmic procedure described in Section IV. After this, the base learner waits until the next data chunk becomes available. After all the base learners are trained with the assigned data chunks and there is no more training data available, SOFEnsemble is ready for classifying unlabeled samples.

For better illustration, the learning procedure of SOFEnsemble is summarized in Algorithm 2.

Algorithm 2 SOFEnsemble Identification.

```
while (a new sample $\boldsymbol{x}_n$ is available) do
  randomly assign $\boldsymbol{x}_n$ to $H$ different data pools;
  for $f = 1$ to $F$ do
    if (the $f^{th}$ data pool is full) then
      pack all the samples in the pool into a data chunk;
      assign the data chunk to the $f^{th}$ SOFIS+;
      update the $f^{th}$ SOFIS+ using Algorithm 1;
      empty the pool;
    end if
  end for
  $n \leftarrow n + 1$;
end while
```

5.2. Validation Policy

During the validation process, for each unlabeled sample $\boldsymbol{x}$, the data distributor will pass it to all F base classifiers . Each base classifier will produce C

scores of confidence corresponding to the C classes using equation (24), and the class label of $\boldsymbol{x}$ is determined as a joint decision of the F base classifiers. In the proposed ensemble framework, the decision fusion module will firstly combine the outputs of the F base classifiers into C overall scores of confidence (one overall score per class):

$$\Lambda^c(\boldsymbol{x}) = \prod_{i=1}^{F} \lambda_i^c(\boldsymbol{x}) \tag{25}$$

where $c = 1, 2, ..., C$; $\Lambda^c(\boldsymbol{x})$ is the product of the F scores of confidence corresponding to the c^{th} class produced by the base classifiers. The rationale behind equation (25) is to enlarge the difference between the respective overall scores of confidence of the C classes for more accurate decision-making.

The final decision is made by equation (26) following the "winner-takes-all" principle:

$$label(\boldsymbol{x}) \leftarrow class\ i^*; \quad i^* \leftarrow \underset{c=1,2,...,C}{\arg\max}\left(\Lambda^c(\boldsymbol{x})\right) \tag{26}$$

However, one may notice that this strategy could be changed to alternative ones, which can be, for example, voting, averaging, without changing the general concept and principles of SOFEnsemble.

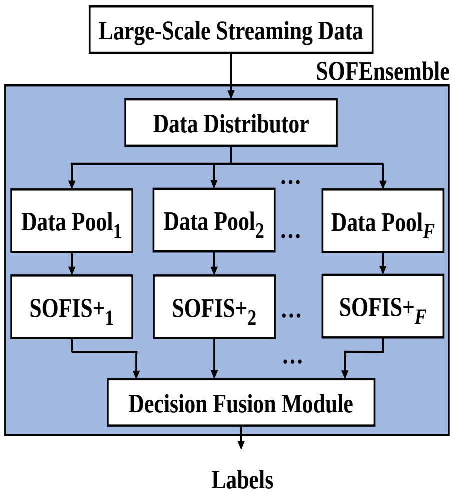


Figure 4: Architecture of SOFEnsemble

6. Experimental Investigation

In this section, numerical experiments based on a wide variety of benchmark datasets are performed for validating the proposed concept and general principles. The large-scale benchmark datasets involved in the experimental studies include: 1) forest covertype (FC); 2) poker hand (POK); 3) skin segmentation (SKIN); 4) SUSY; 5) ECO_E; 6) ECO_CO; 7) EM_E and 8) EM_M. The eight benchmark datasets are available at: `https://archive.ics.uci.edu/ml/index.php`. Following the common practice [39], the FC and POK datasets are further converted to multiple "one-vs-all" binary classification problems by selecting a particular class as the positive one and using the rest as the negative class. To be more specific, for FC dataset, "class 1 vs all", "class 2 vs all", "class 3 vs all" and "class 7 vs all" are considered, and the ratios between majority and minority classes of the four cases are $369,173 : 211,840$, $297,711 : 283,301$, $545,258 : 35,754$ and $560,502 : 20,510$, respectively; for POK dataset, "class 0 vs all" and "class 1 vs all" are considered, and the ratios between majority and minority classes of the two cases are $513,701 : 511,308$ and $591,912 : 433,097$, respectively. In addition, two high-dimensional image classification problems, namely, 9) MNIST (available at: `http://yann.lecun.com/exdb/mnist/`) and 10) Fashion MNIST (FMNIST, available at: `https://github.com/zalandoresearch/fashion-mnist`) are involved for experiments to evaluate the ability of SOFEnsemble on handling high-dimensional, complex problems. Key details of the ten datasets used for numerical examples are summarized in Table 1.

The numerical examples presented in this section are focused on the following two aspects:

1. performance investigation with different externally controlled parameter settings;
2. performance comparison with the state-of-the-art classification approaches.

Unless specifically stated otherwise, the algorithms are developed using MATLAB2018a, and numerical experiments are performed on a Windows10 desktop

Table 1: DETAILS OF DATASETS FOR EXPERIMENT

Dataset	#Samples	#Attributes	#Classes
FC	518,012	54	7
FC_1 (class 1 vs all)	518,012	54	2
FC_2 (class 2 vs all)	518,012	54	2
FC_3 (class 3 vs all)	518,012	54	2
FC_7 (class 7 vs all)	518,012	54	2
POK	1,025,010	10	10
POK_0 (class 0 vs all)	1,025,010	10	2
POK_1 (class 1 vs all)	1,025,010	10	2
SKIN	245,057	3	2
SUSY	5,000,000	18	2
ECO_E	4,178,504	16	10
ECO_CO	4,178,504	16	21
EM_E	4,178,504	16	10
EM_M	4,178,504	16	50
MNIST	70,000	28×28	10
FMNIST	70,000	28×28	10

with dual core Intel Xeon W CPU 4.0×2 GHz and 32 GB RAM. The reported results are obtained after 10 Monte-Carlo experiments. The source codes of SOFIS+ and SOFEnsemble are available at: `https://github.com/Gu-X`.

6.1. Performance Investigation

As aforementioned, SOFEnesemble requires four externally controlled parameters to be determined by users, which include:

1. the level of granularity, G for SOFIS+;
2. the chunk size, L;
3. the number of base learners, F, and;
4. the number of data pools that receive new sample per instance, H.

To investigate the behavior of SOFEnsemble with different parameter settings, in this subsection, numerical experiments based on the following four binary classification problems, namely, FC_1, FC_2, FC_3 and FC_7, are conducted. During the experiments, for each dataset, 80% of the samples are randomly selected out for training and the remaining ones are used for validation. It is worth noting that the majority and minority classes of FC_3 and FC_7 are highly imbalanced, and classifiers may behave differently on highly imbalanced classification problems.

Firstly, the influence of different levels of granularity, G on the classification performance of SOFEnsemble is investigated. In this experiment, the value of G varies from 6 to 10. Other externally controlled parameters are set as: $L = 10,000$, $F = 5$ and $H = 2$. The performance of SOFEnsemble in terms of classification accuracy (Acc), geometric mean (GM) and the average training time consumption for the base learners (t_{exe}, in seconds) are reported in Table 2. The expression of GM is given in equation (27) [12].

$$GM = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}} \tag{27}$$

where TP, TN, FP and FN represent true positive, true negative, false positive and false negative, respectively.

Table 2: CLASSIFICATION PERFORMANCE WITH DIFFERENT LEVELS OF GRANULARITY

G	Measure	FC_1	FC_2	FC_3	FC_7
6	Acc	0.8900	0.8810	0.9751	0.9858
	GM	0.8840	0.8811	0.9698	0.9783
	t_{exe}	40	41	42	43
7	Acc	0.9160	0.9113	0.9811	0.9907
	GM	0.9119	0.9116	0.9733	0.9804
	t_{exe}	55	63	53	53
8	Acc	0.9351	0.9317	0.9859	0.9927
	GM	0.9318	0.9319	0.9730	0.9809
	t_{exe}	87	98	100	107
9	Acc	0.9470	0.9431	0.9886	0.9934
	GM	0.9437	0.9433	0.9724	0.9795
	t_{exe}	154	172	165	170
10	Acc	0.9521	0.9471	0.9895	0.9941
	GM	0.9483	0.9472	0.9715	0.9795
	t_{exe}	217	234	225	232

It can be observed from Table 2 that the classification accuracy of SOFEnsemble is improved given a higher level of granularity because more prototypes are identified during the online learning process. However, this also increases the system complexity, resulting in lower computational efficiency. Trading-off between the classification accuracy and system complexity, the recommended values of G are 8 and 9.

In the next example, the influence of chunk size, L on the classification performance of the proposed ensemble system is studied, where the value of L varies from $2,000$ to $20,000$. Other externally controlled parameters are set as: $G = 9$, $F = 5$ and $H = 2$. The numerical results are reported in Table 3.

Table 3: CLASSIFICATION PERFORMANCE WITH DIFFERENT CHUNK SIZES

L	Measure	FC_1	FC_2	FC_3	FC_7
2,000	Acc	0.9524	0.9471	0.9903	0.9950
	GM	0.9479	0.9472	0.9664	0.9751
	t_{exe}	194	214	72	62
5,000	Acc	0.9500	0.9454	0.9896	0.9941
	GM	0.9461	0.9455	0.9695	0.9773
	t_{exe}	150	171	96	97
10,000	Acc	0.9470	0.9431	0.9886	0.9934
	GM	0.9437	0.9433	0.9724	0.9795
	t_{exe}	154	172	165	170
15,000	Acc	0.9448	0.9410	0.9876	0.9928
	GM	0.9418	0.9412	0.9736	0.9801
	t_{exe}	178	199	211	218
20,000	Acc	0.9431	0.9397	0.9868	0.9923
	GM	0.9403	0.9399	0.9743	0.9809
	t_{exe}	199	220	248	259

One can see from this table that the value of L has a small impact on both the classification accuracy and computational efficiency. The proposed SOFEnsemble can perform classification with a higher accuracy rate if a smaller chunk size is used. This is because that a small chunk size improves the sensitivity of the learning system to capture the shifts and/or drifts in the data patterns of the data stream. Meanwhile, this also increases the overall computational complexity because the fusion center of the ensemble model has to update the knowledge bases more frequently. Based on Table 3, the recommended value range of L is between 5,000 and 15,000.

Then, the classification performance of SOFEnsemble with different numbers of base classifiers, F is investigated. In this example, the value of F varies from 3 to 8, and other parameters are fixed as: $G = 9$, $L = 10,000$ and $H = 2$. The results are tabulated in Table 4, where it can be observed that the overall training time consumption reduces by increasing the number of base learners because each base learner now receives less training samples from the data distributor. Meanwhile, each base learner tends to make more mistakes during the classification stage due to the insufficient amount of training samples, resulting in lower overall classification accuracy rate by SOFEnsemble. In addition, the proposed ensemble model requires more computational resources with the increase of ensemble scale. Considering the computational efficiency, overall classification accuracy and computational resource consumption of the proposed SOFEnsemble model, $F = 5$ is used for the remaining numerical examples of this section.

In the last example of this subsection, the influence of the number of data pools receiving new sample per instance, H on the classification performance of SOFEnsemble is studied. In this study, the value of H varies from 1 to 5. Other externally controlled parameters are set as $G = 9$, $L = 10,000$ and $F = 5$. Note that when $H = 1$, the data distributor of SOFEnsemble assigns every training sample to only one base learner, which is the same as the majority of the existing ensemble frameworks [31, 32, 11]. If $H = 5$, all the base classifiers will receive the same training samples and lose the diversity. In such case, the decisions made by SOFEnsemble during the classification process will be exactly the same as any one of its base learners. Thus, the performance of SOFEnsemble is equivalent to its single-model base classifier, SOFIS+. The results obtained in the experiments are presented in Table 5. As one can see from this table, the commonly-used data distribution policy (namely, $H = 1$) performs the worst in terms of classification accuracy (even worse than using a single SOFIS+), though it significantly improves the computational efficiency of the ensemble model. SOFEnsemble achieves the highest classification precision if $H = 4$, and its computational efficiency is still higher than the single-model SOFIS+ (namely, $H = 5$) . Therefore, one may conclude from Table 5 that the proposed data distribution policy not only helps SOFEnsemble to perform more accurate

Table 4: CLASSIFICATION PERFORMANCE WITH DIFFERENT NUMBERS OF BASE CLASSIFIERS

F	Measure	FC_1	FC_2	FC_3	FC_7
3	Acc	0.9490	0.9462	0.9895	0.9942
	GM	0.9467	0.9463	0.972	0.9817
	t_{exe}	345	394	294	296
4	Acc	0.9480	0.9448	0.9889	0.9937
	GM	0.9452	0.9449	0.9718	0.9804
	t_{exe}	218	251	212	218
5	Acc	0.9470	0.9431	0.9886	0.9934
	GM	0.9437	0.9433	0.9724	0.9795
	t_{exe}	154	172	165	170
6	Acc	0.9453	0.9408	0.9879	0.9932
	GM	0.9416	0.9410	0.9707	0.9783
	t_{exe}	118	133	135	140
7	Acc	0.9445	0.9397	0.9876	0.9929
	GM	0.9404	0.9399	0.9701	0.9756
	t_{exe}	95	106	114	119
8	Acc	0.9432	0.9381	0.9870	0.9927
	GM	0.9385	0.9383	0.9689	0.9750
	t_{exe}	78	87	98	104

Table 5: CLASSIFICATION PERFORMANCE WITH DIFFERENT NUMBERS OF DATA POOLS RECEIVING NEW SAMPLE PER INSTANCE

H	Measure	FC_1	FC_2	FC_3	FC_7
1	Acc	0.9377	0.9320	0.9860	0.9919
	GM	0.9326	0.9322	0.9666	0.9708
	t_{exe}	60	64	80	80
2	Acc	0.9470	0.9431	0.9886	0.9934
	GM	0.9437	0.9433	0.9724	0.9795
	t_{exe}	154	172	165	170
3	Acc	0.9503	0.9468	0.9895	0.9942
	GM	0.9478	0.9469	0.9742	0.9821
	t_{exe}	296	347	264	275
4	Acc	0.9506	0.9476	0.9898	0.9946
	GM	0.9485	0.9477	0.9739	0.9829
	t_{exe}	464	550	362	365
5	Acc	0.9455	0.9430	0.9892	0.9937
	GM	0.9435	0.9430	0.9690	0.9785
	t_{exe}	660	803	463	461

classification, but also effectively reduces the training time consumption. Based on this numerical example, the best values of H are 2 and 3.

6.2. Performance Comparison

In this subsection, the performance of the proposed ensemble model is compared with alternative approaches including popular single-model classification methods and the state-of-the-art ensemble models designed for big data classification. The following externally controlled parameter setting of SOFEnsemble, namely, $G = 9$, $L = 10,000$, $F = 5$ and $H = 2$ are used in numerical examples for consistency. It has to be stressed that the main focus of this paper is for demonstrating the proposed concept, and only a general setting is used for experimental investigation. However, the best parameter setting may differ from case to case depending on the nature of the problems and the availability of computational resources.

Firstly, the performance of SOFEnsemble is compared with nine popular zero-order and first-order EISs on the four binary classification problems used in the previous numerical examples, namely, FC_1, FC_2, FC_3 and FC_7 datasets under the same experimental setting. The nine EISs used for benchmark comparison are as follows: 1) SOFIS [13]; 2) eTS [9]; 3) SAFIS [18]; 4) eClass0 [19]; 5) eClass1 [19]; 6) ESAFIS [46]; 7) ALMMo0 [47]; 8) ALMMo [16] and 9) PALM [30]. During numerical experiments, SOFIS uses Euclidean distance, and the level of granularity, G is set to be 9; eTS, SAFIS, eClass0, eClass1, ESAFIS, ALMMo0, ALMMo and PALM follow the recommended parameter settings given by [9, 18, 19, 46, 47, 16, 30]. Note that SOFIS, eClass0 and ALMMo0 are zero-order EISs that are mostly used for classifications. eTS, SAFIS, eClass1, ESAFIS, ALMMo and PALM are first-order EISs. Among them, eClass1 is designed for classification only, while eTS, SAFIS, ESAFIS, ALMMo and PALM are primarily designed for regression, but can be used for binary classification as well. ESAFIS and ALMMo are also suitable for multi-class classification. Classification performance of SOFEnsemble and nine EIS counterparts are reported in Table 6. The performance of SOFIS+ is also given by the same table as the baseline. One can see from Table 6 that SOFEnsemble outperforms its counterparts in terms of the three performance measures, namely, classification accuracy, geometric mean and training time consumption in the vast majority of cases. This demonstrates the superiority of the proposed ensemble framework over alternative single-model EIS approaches. From Table 6, one may also notice that, typically, zero-order EISs perform better than first-order ones on classification tasks.

Secondly, the performance of SOFEnsemble is compared with a wide variety of classification methods on FC, MNIST and FMNIST datasets, which include: 1) SOFIS [13]; 2) ESAFIS [46]; 3) eClass0 [19]; 4) eClass1 [19]; 5) ALMMo0

Table 6: CLASSIFICATION PERFORMANCE COMPARISON WITH POPULAR EIS APPROACHES

Dataset	FC_1			FC_2			FC_3			FC_7		
Algorithm	*Acc*	*GM*	t_{exe}	*Acc*	*GM*	t_{exe}	*Acc*	*GM*	t_{exe}	*Acc*	*GM*	t_{exe}
SOFEnsemble	**0.9470**	**0.9437**	**154**	**0.9431**	**0.9433**	**172**	0.9886	**0.9724**	**165**	0.9934	0.9795	**170**
SOFIS+	0.9455	0.9435	660	0.9430	0.9430	803	**0.9892**	0.9690	463	0.9937	0.9785	461
SOFIS	0.9086	0.9075	2166	0.9060	0.9062	4002	0.9787	0.9721	3027	0.9891	**0.9838**	2930
eTS	0.6354	0.0000	4516	0.5124	0.0000	4729	0.9385	0.0000	3891	0.9647	0.0000	3410
SAFIS	0.6272	0.3206	1375	0.5446	0.2804	1191	0.9386	0.0288	263	0.9649	0.0527	199
eClass0	0.6156	0.6062	1538	0.5939	0.5939	1446	0.9185	0.8212	2380	0.8064	0.7715	2381
eClass1	0.6354	0.0000	632	0.5124	0.0000	628	0.9385	0.0000	590	0.9647	0.0000	617
ESAFIS	0.8149	0.7891	42253	0.8002	0.8004	37115	0.9696	0.8677	14415	0.9819	0.7981	22820
ALMMo0	0.9348	0.9290	4278	0.9265	0.9265	4274	0.9870	0.9401	8060	**0.9944**	0.9548	8346
ALMMo	0.7662	0.7325	329	0.7544	0.7544	322	0.9594	0.8681	327	0.9700	0.4245	384
PALM	0.7672	0.7351	284	0.7552	0.7555	285	0.9596	0.8652	288	0.9679	0.3502	296

[47]; 6) ALMMo [16]; 7) k-nearest neighbors classifier (KNN) [42]; 8) SVM [4]; 9) decision tree (DT) [43]; 10) RF [5]; 11) sequential classifier (SC) [44]; 12) multi-layer perceptron (MLP); 13) extreme learning machine (ELM) [45]; and 14) hierarchical prototype-based classifier (HP) [48]. During numerical experiments, k is set to be 10 for KNN; SVM uses Gaussian kernel; the ensemble model of RF is composed of 200 classification trees; MLP has three hidden layers, each layer has 20 neurons; the maximum number of neurons for ELM is set to be 500; and the layer number of HP is set as 6. For FC dataset, half of the data samples are randomly selected out to form the training set, and the remaining samples are used for validation. For MNIST and FMNIST, the Gist feature descriptor [41] is employed to extract a 512×1 dimensional feature vector from each image for training and testing. The original split for training and testing sets is kept, but the order of training samples is randomly scrambled. Classification performances of the 15 approaches in the form of Acc and t_{exe} on the three datasets are presented in Table 7, where the training time of KNN is not reported because the algorithm requires no training. It can be observed from Table 7 that SOFEnsemble is able to achieve good performance on the three problems with high computational efficiency, outperforming the majority of the comparative algorithms in terms of both classification accuracy and training time consumption.

For better illustration, pairwise Wilcoxon tests between SOFEnsemble and the selected comparative approaches with better classification performance including, SOFIS, KNN, RF, SC, ELM, ALMMo0 and HP are conducted to demonstrate statistical significance of the classification performance of SOFEnsemble over the alternative prediction models. The returned p-values from the hypothesis test during a particular experiment are tabulated in Table 8, where one can see from the p-values that the null hypothesis is rejected in the majority of cases, suggesting that SOFEnsemble outperforms the best performing comparative approaches, statistically.

Next, the performance of SOFEnsemble is compared with two state-of-the-art image classification methods, namely, 1) deep conventional neural network

Table 7: CLASSIFICATION PERFORMANCE COMPARISON WITH POPULAR CLASSIFICATION APPROACHES

Dataset	FC		MNIST		FMNIST	
Algorithm	*Acc*	t_{exe}	*Acc*	t_{exe}	*Acc*	t_{exe}
SOFEnsemble	0.9173	192	**0.9869**	53	**0.9017**	54
SOFIS	0.8778	1757	0.9866	956	0.8868	1085
ESAFIS	0.7359	5480	0.9830	34612	0.8962	29388
eClass0	0.3456	74	0.8386	100	0.7351	95
eClass1	0.3647	374	0.9764	10769	0.8878	10786
ALMMo0	0.8932	1717	0.9864	462	0.8882	470
ALMMo	0.7012	160	0.9748	5627	0.8735	4881
KNN	0.9107	-	0.9852	-	0.8966	-
SVM	0.7247	4641	0.9857	153	0.8936	178
DT	0.9180	38	0.9010	196	0.8088	153
RF	**0.9591**	715	0.9631	4191	0.8872	4501
SC	0.8472	17137	0.9762	1111	0.8817	1139
MLP	0.7667	214	0.6011	80	0.8289	102
ELM	0.6481	**14**	0.9424	**3**	0.8465	**3**
HP	0.9069	545	0.9864	76	0.8845	99

Table 8: p-VALUES IN PAIRWISE WILCOXON TESTS

	Algorithm	FC	MNIST	FMNIST
SOFEnsemble vs	SOFIS	0.0000	0.2898	0.0043
	ALMMo0	0.0000	0.2840	0.0877
	KNN	0.0000	0.0171	0.0000
	RF	0.0000	0.0003	0.1055
	SC	0.0000	0.1955	0.9464
	ELM	0.0000	0.0000	0.0000
	HP	0.0000	0.7711	0.0290

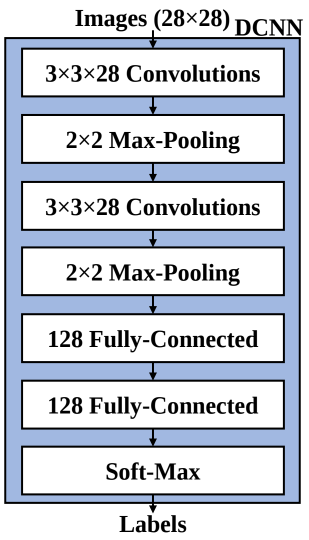


Figure 5: Architecture of DCNN for image classification.

Table 9: CLASSIFICATION PERFORMANCE COMPARISON WITH DCNN AND DRB ON IMAGE CLASSIFICATION PROBLEMS

Algorithm	MNIST	FMNIST
SOFEnsemble	**0.9918**	**0.9095**
DRB	0.9914	0.9004
DCNN	0.9913	0.9078

(DCNN) [49] and 2) DRB system [32] on MNIST and FMNIST datasets. In this example, a seven-layer DCNN with the architecture given by Fig. 5 is trained to classify images of the two images sets [49]. The DCNN is implemented using the Keras module from Tensorflow and the network training is conducted on a Windows10 laptop with a NVIDIA GeForce RTX 2070 GPU. Following the experimental setting of [49], the DCNN uses the categorical cross-entropy loss function and the adaptive moment estimation algorithm [50] as the optimizer. For network training, 90% of images from the training set are randomly selected out for training and the remaining images in the training set are used for validation. The DCNN is used for classifying testing images after 40 training epochs and the average accuracy rates are reported in Table 9. The trained DCNN is then employed by DRB as its feature descriptor. 128×1 dimensional activations from the last fully-connected layer are used as the feature vectors of the images. For fair comparison, SOFEnsemble uses the same feature vectors extracted by the feature descriptor of DRB for training and testing. The classification accuracy rates of SOFEnsemble and DRB on testing images of MNIST and FMNIST are also reported in the same table. From Table 9 one can see that SOFEnsemble surpasses both DCNN and DRB in terms of classification accuracy on both image sets.

In the following numerical examples, the classification performance of SOFEnsemble is further compared with the state-of-the-art fuzzy approaches designed for handling big data problems, which include:

1. Chi-FRBCS-BigDataCS [38];
2. Chi-FRBCS-BigDataGlobal [39];
3. Chi-FRBCS-BigData [35];
4. Fuzzy binary decision tree (FBDT) [40], and;
5. Fuzzy multi-way decision tree (FMDT) [40].

Parameter settings of these approaches and computational devices used for numerical examples are given in Table 10.

The performance of SOFEnsemble is firstly compared with Chi-FRBCS-BigDataCS [38] and Chi-FRBCS-BigDataGlobal [39] on the following large-scale binary classification problems: FC_1, FC_2, FC_3, FC_7, POK_0, POK_1, SKIN and SUSY. During the experiments, for each dataset, 80% of the samples are randomly selected out for training and the remaining ones are used for validation. Classification performances demonstrated by the three approaches are reported in Table 11 in terms of geometric mean (GM) and t_{exe}. The best results are in bold.

For better evaluating the ability of handling large-scale datasets, SOFEnsemble is then compared with Chi-FRBCS-BigData [35], FBDT [40] and FMDT [40] on the following large-scale problems: ECO_E, ECO_CO, EM_M, EM_M, POK and SUSY. The same experimental protocol as used in Table 11 is adopted. Statistical performances of the four approaches obtained on the six benchmark datasets in terms of Acc and t_{exe} ($mean \pm standard\ deviation$) are tabulated in Table 12, where the best results are in bold.

It can be observed from Tables 11 and 12 that SOFEnsmble is able to achieve very high classification accuracy rates on these large-scale problems surpassing or, at least, on par with the state-of-the-art fuzzy classifiers for big data. In addition, the training time consumption of the proposed ensemble model is also very low despite that numerical experiments are performed on a Windows10 desktop. This indicates that the computational efficiency of SOFEnsemble can be further improved if a cluster is used, like the works in [39, 40].

Table 10: PARAMETER SETTINGS AND COMPUTATIONAL DEVICES FOR NUMERICAL EXPERIMENTS

Algorithm	Parameter Setting	Computational Resources
SOFEnsemble	$G = 9$, $L = 10,000$, $F = 5$, and $H = 2$	A Windows10 desktop (Intel Xeon W CPU with dual core at 4.0 GHz, 32GBRAM)
Chi-FRBCS-BigDataCS [38]	Inference= WinningRule, ruleWeight=PCF, NumFuzzyLabels=3 [39]	A CentOS cluster with one master node (Intel Xeon E5 CPU with quad core at 2.4 GHz, 8 GB RAM), four slave nodes with two Intel Xeon E5 CPUs at 2.4 GHz with six cores in each, 32 GB RAM, and another three slave nodes with two Intel Xeon E5 CPUs at 2.1 GHz with six cores in each, 32 GB RAM [39].
Chi-FRBCS-BigDataGlobal [39]	Inference= WinningRule, ruleWeight=PCF, NumFuzzyLabels=3, NumRuleSubsets=4, MinOccFreqSubsets=10, MaxRuleperReducer=400000 [39]	
Chi-FRBCS-BigData [35]	Inference= WinningRule, ruleWeight=PCF, NumFuzzyLabels=3 [40]	A Linux cluster with one master node (quad core Intel i5 CPU 2.67GHz,8GB RAM) and three slave nodes (each is equipped with quad core Intel i7 CPU 3.40GHz, and 16GB RAM)[40]
FBDT [40]	$\gamma = 0.1\%, \phi = 1, \lambda = 1$ [40]	
FMDT [40]	$\gamma = 0.1\%, \phi = 0.02N$, $\lambda = 10^{-4}$ [40]	

Table 11: PERFORMANCE COMPARISON BETWEEN SOFENSEMBLE, CHI-FRBCS-BIGDATAGLOBAL AND CHI-FRBCS-BIGDATACS ON LARGE-SCALE BINARY CLASSIFICATION PROBLEMS [39]

Algorithm	Measure	FC_1	FC_2	FC_3	FC_7	POK_0	POK_1	SKIN	SUSY
SOFEnsemble	GM	**0.9595**	**0.9433**	**0.9724**	**0.9795**	**0.7018**	**0.6172**	**0.9996**	**0.7119**
	T_{exe}	**22**	172	165	170	264	271	**13**	994
Chi-FRBCS-BigDataGlobal [39]	GM	0.7531	0.7291	0.9565	0.9281	0.6336	0.5848	0.9597	0.5524
	T_{exe}	76	75	74	75	107	110	53	**103**
Chi-FRBCS-BigDataCS [38]	GM	0.7528	0.7296	0.9551	0.9089	0.6183	0.5616	0.9595	0.5477
	T_{exe}	68	**70**	**69**	**70**	**58**	**59**	22	1359

Table 12: PERFORMANCE COMPARISON BETWEEN SOFENSEMBLE, FMDT, FBDT AND CHI-FRBCS-BIGDATA ON LARGE-SCALE CLASSIFICATION PROBLEMS [40]

Algorithm	Measure	ECO_E	ECO_CO	EM_E
SOFEnsemble	*Acc*	0.**9888±0.0002**	**0.9765±0.0003**	**0.9747±0.0003**
	T_{exe}	1257	783	823
Chi-FRBCS-BigData [35]	*Acc*	0.5449±0.0781	0.7360±0.0641	0.6728±0.0895
	T_{exe}	1263	1491	1276
FBDT [40]	*Acc*	0.7824±0.0004	0.9780±0.0002	0.9693±0.0003
	T_{exe}	720	1070	603
FMDT [40]	*Acc*	0.9759±0.0004	0.9753±0.0002	0.9691±0.0002
	T_{exe}	**392**	**779**	**372**
Algorithm	Measure	EM_M	POK	SUSY
SOFEnsemble	*Acc*	**0.9735±0.0002**	0.6258±0.0006	0.7525±0.0004
	T_{exe}	**2259**	314	994
Chi-FRBCS-BigData [35]	*Acc*	0.9271±0.0760	0.0518±0.0005	0.5575±0.0016
	T_{exe}	62175	18918	1444
FBDT [40]	*Acc*	0.9675±0.0002	0.6248±0.0050	**0.7972±0.0004**
	T_{exe}	3916	11	**255**
FMDT [40]	*Acc*	0.9600±0.0003	**0.7718±0.0007**	0.7964±0.0002
	T_{exe}	4005	**3**	**255**

6.3. Remarks

Numerical examples given in this section justify the efficacy of the proposed approach as a powerful tool for handling large-scale data streams. There are a few remarks worth noting.

Firstly, one has to admit that a universally best classification algorithm does not exist. The performance of a particular classification algorithm depends on many factors such as, its inherent learning mechanism, the nature of data, the available computational resources. In addition, with the widely deployment of artificial intelligence technologies, the explainability and interpretablity of machine learning algorithms is becoming increasingly important, especially for safety-critical application scenarios. Generally speaking, DT, ELM, KNN, MLP, RF and SVM are the state-of-the-art approaches for static data classification. These approaches have demonstrated very attractive results on many challenging problems, but most of them are applicable to offline application scenarios only and their explainability is very limited. DCNN is the dominant approach for image classification, but is also a well-known "black box" type model. In addition, GPUs are needed to facilitate DCNN training. In contrast, EISs are the most popular approaches for handling streaming data, offering higher model-transparency and interpretablity than the vast majority of the state-of-the-art approaches. First-order EISs are designed primarily for regression and only few of them can be used for multi-class classification directly without modification. First-order EISs are very efficient on low dimensional problems, but their computational efficiency decreases significantly on high dimensional problems due to the recursive updating of covariance matrices. Zero-order EISs are designed for classification, and they usually perform better than first-order EISs on classification tasks. Nevertheless, zero-order EISs often struggle when dealing with non-linear problems and they also suffer from system obesity when handling large-scale, complex problems.

As an ensemble model designed for large-scale streaming data classification, SOFEnsemble is able to achieve greater classification precision with much higher computational efficiency. Thanks to the prototype-based nature of its base learners, namely, SOFIS+, the model structure of SOFEnsemble is highly transparent and its decision-making process is fully explainable. Very importantly, SOFEnsmble is less likely to experience system obesity as other single-model EISs thanks to its parallel ensemble architecture. Although SOFEnsemble is capable of constructing more precise classification boundaries based on identified prototypes by considering both the inter-class and intra-class distances between them, its classification precision may significantly decrease if the data structure is over complex and samples of different classes are not linearly separable, same as other zero-order EISs. In addition, to fully appreciate the strength of SOFEnsemble, specialized computational equipments are needed. The implementation cost of SOFEnsemble is much higher than single-model EISs. Therefore, a trade-off between implementation cost and performance has to be considered in real-world applications.

7. Conclusion

In this paper, a novel fuzzy ensemble classifier named SOFEnsemble is introduced for large-scale problems. The proposed ensemble system is built upon a number of SOFIS+ that learn from streaming data on a chunk-by-chunk basis and continuously self-update the decision boundaries by identifying the more representative samples. Numerical examples based on large-scale, complex problems show that SOFEnsemble is able to outperform, or at least, on par with the state-of-the-art classification approaches in terms of both classification accuracy and computational efficiency, justifying the validity of the proposed ensemble framework. There are several considerations for future work. Firstly, the optimality of the prototypes identified from data needs to be investigated. Due to the "one pass" learning procedure of SOFIS+, one may consider further optimizing the learned prototypes by SOFIS+. It is expected that SOFIS+ can build more precise decision boundaries and achieve better classification performance through prototype optimization. Secondly, the computational efficiency of SOFEnsemble needs to be further improved. In the current version, SOFIS+

learns from each data chunk sample-by-sample, the computational efficiency may decrease significantly if the chunk size is too large. It may be worth modifying the learning procedure of SOFIS+ so that the system can learn from data in a more efficient manner. On the other hand, one may notice that SOFEnsemble is developed on Matlab platform and implemented on a single desktop at this moment. A boost in computational efficiency can be expected if SOFEnsemble is implemented on a cluster using Python or C++. Thirdly, one may also consider to involve some feature selection techniques to reduce the computational complexity and increase the diversity between different base learners.

References

[1] X. Gu, Q. Shen, and P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, DOI: 10.1109/TCYB.2020.2967462, 2020.

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

[3] T. Kohonen, "Learning vector quantization," in *Self-Organizing Maps*, Berlin, Heidelberg: Springer, 1995, pp. 175–189.

[4] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods.* Cambridge: Cambridge University Press, 2000.

[5] L. Breiman, "Random forests," *Mach. Learn. Proc.*, vol. 45, no. 1, pp. 5–32, 2001.

[6] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep learning for remote sensing image classification: a survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. e1264, 2018.

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

[8] D. Leite, I. Škrjanc, and F. Gomide, "An overview on evolving systems and learning from stream data," *Evol. Syst.*, DOI: 10.1007/s12530-020-09334-5, 2020.

[9] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst. Man, Cybern. - Part B Cybern.*, vol. 34, no. 1, pp. 484–498, 2004.

[10] H. Hagras, "Toward human-understandable, explainable AI," *Computer (Long. Beach. Calif).*, vol. 51, no. 9, pp. 28–36, 2018.

[11] M. Pratama, W. Pedrycz, and E. Lughofer, "Evolving ensemble fuzzy classifier," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2552–2567, 2018.

[12] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: a survey," *Inf. Fusion*, vol. 37, pp. 132–156, 2017.

[13] X. Gu and P. P. Angelov, "Self-organising fuzzy logic classifier," *Inf. Sci. (Ny).*, vol. 447, pp. 36–51, 2018.

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

[15] G. Leng, G. Prasad, and T. M. McGinnity, "An on-line algorithm for creating self-organizing fuzzy neural networks," *Neural Networks*, vol. 17, no. 10, pp. 1477–1493, 2004.

[16] P. P. Angelov, X. Gu, and J. C. Principe, "Autonomous learning multimodel systems from data streams," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 2213–2224, 2018.

[17] H. Han and J. Qiao, “A self-organizing fuzzy neural network based on a growing-and-pruning algorithm,” *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 6, pp. 1129–1143, 2010.

[18] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, “Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction,” *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[19] P. Angelov and X. Zhou, “Evolving fuzzy-rule based classifiers from data streams,” *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1462–1474, 2008.

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

[21] J. De Jesús Rubio, “SOFMLS: online self-organizing fuzzy modified least-squares network,” *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[22] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, “PANFIS: a novel incremental learning machine,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 55–68, 2014.

[23] M. Pratama, S. G. Anavatti, and E. Lughofer,“Genefis: toward an effective localist network,” *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 3, pp. 547–562, 2014.

[24] K. Subramanian, A. K. Das, S. Sundaram, and S. Ramasamy, “A meta-cognitive interval type-2 fuzzy inference system and its projection based learning algorithm,” *Evolving Systems*, vol. 5, no. 4, pp. 219–230, 2014.

[25] M. Pratama, J. Lu, and G. Zhang, “Evolving type-2 fuzzy classifier,” *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 3, pp. 574–589, 2016.

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

[27] 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.

[28] H. Rong, Z. Yang, and P. K. Wong, "Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2019.2931871, 2019.

[29] D. Ge and X. J. Zeng,"A self-evolving fuzzy system which learns dynamic threshold parameter by itself," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 8, pp. 1625-1637, 2019.

[30] M. Ferdaus, M. Pratama, S. Anavatti, and M. Garratt, "PALM: an incremental construction of hyperplanes for data stream regression," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 11, pp. 2115–2129, 2019.

[31] J. A. Iglesias, A. Ledezma, and A. Sanchis, "Ensemble method based on individual evolving classifiers," in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2013, pp. 56–61.

[32] X. Gu, P. P. Angelov, C. Zhang, and P. M. Atkinson, "A massively parallel deep rule-based ensemble classifier for remote sensing scenes," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 345–349, 2018.

[33] E. Soares, P. Costa, B. Costa, and D. Leite, "Ensemble of evolving data clouds and fuzzy models for weather time series prediction," *Appl. Soft Comput.*, vol. 64, pp. 445–453, 2018.

[34] V. López, S. del Río, J. M. Benítez, and F. Herrera, "On the use of MapReduce to build linguistic fuzzy rule based classification systems for big data," in IEEE International Conference on Fuzzy Systems, 2014, pp. 1905–1912.

[35] S. del Río, V. López, J. M. Benítez, and F. Herrera, "A MapReduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules," *Int. J. Comput. Intell. Syst.*, vol. 8, no. 3, pp. 422–437, 2015.

[36] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[37] Z. Chi, H. Yan, and T. Pham, "Fuzzy algorithms: with applications to image processing and pattern recognition". Singapore: World Scientific., 1996.

[38] V. López, S. Del Río, J. M. Benítez, and F. Herrera, "Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data," *Fuzzy Sets Syst.*, vol. 258, pp. 5–38, 2015.

[39] M. Elkano, M. Galar, J. Sanz, and H. Bustince, "CHI-BD: a fuzzy rule-based classification system for big data classification problems," *Fuzzy Sets Syst.*, vol. 348, pp. 75–101, 2018.

[40] A. Segatori, F. Marcelloni, and W. Pedrycz, "On distributed fuzzy decision trees for big data," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 1, pp. 174–192, 2018.

[41] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[42] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers," *Mult. Classif. Syst.*, vol. 34, pp. 1–17, 2007.

[43] L. Lu, L. Di, and Y. Ye, "A decision-tree classifier for extracting transparent plastic-mulched Landcover from landsat-5 TM images," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 7, no. 11, pp. 4548–4558, 2014.

[44] R. N. Patro, S. Subudhi, P. K. Biswal, and F. Dell'Acqua, "Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data," *Int. J. Remote Sens.*, vol. 40, no. 13, pp. 4996–5024, 2019.

[45] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 2, pp. 513–529, 2012.

[46] H. J. Rong, N. Sundararajan, G. Bin Huang, and G. S. Zhao, "Extended sequential adaptive fuzzy inference system for classification problems," *Evol. Syst.*, vol. 2, no. 2, pp. 71–82, 2011.

[47] P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (ALMMo-0)," in *IEEE International Conference on Evolving and Autonomous Intelligent Systems*, 2017, pp. 1–7.

[48] X. Gu and W. Ding, "A hierarchical prototype-based ap proach for classification," *Inf. Sci. (Ny).*, vol. 505, pp. 325–351, 2019.

[49] S. Bhatnagar, D. Ghosal, and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," in *International Conference on Image Information Processing*, 2017, pp. 357–362.

[50] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations*, 2015, pp. 1–15.