An Explainable Semi-Supervised Self-Organizing Fuzzy Inference System for Streaming Data Classification

Xiaowei Gu

School of Computing, University of Kent, Canterbury, CT2 7NZ, UK
E-mail: X.Gu@kent.ac.uk

Abstract: As a powerful tool for data streams processing, the vast majority of existing evolving intelligent systems (EISs) learn prediction models from data in a supervised manner. However, high-quality labelled data can be difficult to obtain in many real-world classification applications concerning data streams, though unlabelled data is plentiful. To overcome the labelling bottleneck and construct a stronger classification model, a novel semi-supervised EIS is proposed in this paper. After being primed with a small amount of labelled data, the proposed method is capable of continuously self-developing its system structure and self-updating the meta-parameters from unlabelled data streams chunk-by-chunk in a non-iterative, exploratory manner by exploiting a novel pseudo-labelling strategy. Thanks to its transparent prototype-based structure and human-understandable reasoning process, the proposed method can provide users high explainability and interpretability while achieving great classification precision. Experimental investigation demonstrates the superior performance of the proposed method.

Keywords: data stream classification; evolving intelligent system; semi-supervised learning; pseudo-labelling;

1. Introduction

Evolving intelligent systems (EISs) [1], [2] are a class of fuzzy systems designed to self-develop and self-evolve from data streams online to dynamically model nonstationary problems in real time. EISs are capable of continuously learning from streaming data "on the fly" in a single-pass manner and transforming newly learned knowledge into human-interpretable IF-THEN fuzzy rules. They can capture the concept shifts and drifts in data streams by self-evolving both the system structure and parameters in accordance with changing data patterns [3]. Thanks to the highly transparent system structure and explainable fuzzy reasoning scheme, EISs are widely recognized as a powerful tool for streaming data processing, by offering both great prediction precision and high model interpretability. Hence, EISs have become increasingly popular since the underlying concept was firstly introduced two decades ago [1], [2]. To date, there have been a variety of successful EISs proposed for solving data stream classification and regression problems. The most representative EISs include, but are not limited to, eTS [1], DENFIS [2], SOFNN [4], eClass [5], SAFIS [6], GENEFIS [7], FLEXFIS [8], PANFIS [9], SOFIS [10], ALMMo [11], and LEOA [12]. Interested readers are referred to the recent survey papers [13]–[15] for more details about the latest developments of EISs and their applications.

Despite that different evolving schemes for structure learning and parameter updating may be employed, the vast majority of existing EISs assemble the prediction models from streaming data in a fully supervised manner. However, in many real-world classification problems, high-quality labelled data is scarce and can be very expensive to acquire due to the high cost of manual labelling [16]. On the other hand, unlabelled data is plentiful, but supervised learning methods, including conventional EISs and mainstream classifiers, such as support vector machine (SVM) [17], *k*-nearest neighbour (*k*NN) [18], decision tree (DT) [19], artificial neural network (ANN) [20], etc., cannot utilize them in training. It is well known that synthetic data augmentation techniques [21], [22] can be employed to greatly increase the amount of labelled data and, hence, improve the generalization ability of the classification models. However, the statistical characterization given by limited data is usually poor. Oversampling a small set of labelled samples for training often makes the learned models more vulnerable to uncertainties and less robust [23].

Semi-supervised learning technique is a hybridization between supervised learning and unsupervised learning [16]. It overcomes the labelling bottleneck by utilizing a great amount of unlabelled data together with labelled ones to build more precise classification models. Existing semi-supervised learning methods can be broadly categorized into two major groups: 1) inductive methods and 2) transductive methods [16]. The primary goal of inductive methods, e.g., semi-supervised SVM (S^3VM) [24], safe semi-supervised SVM (S^4VM) [25], self-training [26], [27], co-training [28], etc., is to construct classification models from both labelled and unlabelled samples that can be used for classifying any data samples in the data space. Transductive methods, which include

local and global consistency (LGC) [29], Laplacian SVM (LapSVM) [30] and anchor graph regularization (AGR) [31], etc., aim to classify only these unlabelled samples presented during training without constructing classification models.

Self-training [26], [27] is a simple, yet effective inductive semi-supervised learning method enjoying the most popularity among alternative ones. The core idea behind self-training is the so-called "pseudo labelling" [32]. A standard self-training approach firstly builds a classification model, which usually can be *k*NN [33], SVM [34], DT [35] or other mainstream classifiers, with labelled data. Then, the classification model is re-trained iteratively by selecting these unlabelled samples with the highest classification confidence to augment the labelled training set together with their respective predicted labels, namely, pseudo labels. Although self-training offers an easy-to-implement solution to construct a strong classifier with minimum human supervision, conventional self-training methods as well as alternative semi-supervised learning methods are limited to offline applications due to the requirement of iterative computation. Iterative computation inevitably causes the propagation of pseudo-labelling errors, deteriorating classification performance. It also unfavourably impairs the interpretability of the self-training process and the transparency of the learned model.

There have been a few self-training methods proposed more recently that are designed to self-construct a highly transparent classification model from streaming data. For example, in [36], a self-training EIS called semi-supervised deep rule-based (SSDRB) classifier is proposed for image stream classification. SSDRB is capable of self-learning a precise classifier from unlabelled streaming images sample-by-sample after being primed with a small amount of labelled images. SSDRB is also able to recognize images with unfamiliar data patterns that are unseen before. The self-training mechanism of SSDRB is further improved in [26], where the classification model learns from unlabelled images on a chunk-by-chunk basis, leading to higher computational efficiency and greater precision. A weakly-supervised self-evolving deep neural network for data stream classification named parsimonious network (ParsNet) is introduced in [37]. A key feature of ParsNet is the so-called self-labelling with hedge method. This method helps the network to address the accumulation of pseudo-labelling errors through a regularization strategy that controls the amount of information to be accepted from pseudo labels. In [38], a skip-connected evolving recurrent network (SERN) is proposed for self-training from unlabelled streaming data. SERN uses an auto-learned mapping function to assign pseudo labels to unlabelled data. Upon availability of true data labels, SERN will compare the produced pseudo labels with the true labels and penalizes the model by resetting its parameters to an earlier point if label mismatches are observed. A self-evolving mutually-operative recurrent network-based model (SERMON) is proposed in [39] for online machine fault monitoring in label delay scenarios. SERMON is a hybrid model composed of *i)* a label mapping unit (MU), *ii)* SERN, and *iii)* a multilayer evolving recurrent network (MERN). During operation, MU supervises SERN to self-learn from unlabelled data by providing pseudo labels, while MERN only learns from labelled data. Different from [38], the regularization strategy used by SERN in [39] allows the model to reset its parameters to the point where the pseudo-labelling error firstly occurs. A generic online semi-supervised learning method, which provides a high level of transparency and human-interpretability, named self-training hierarchical prototype-based (STHP) classifier, is proposed in [27]. STHP can initialize its multi-layered prototype-based structure with a small set of labelled data from scratch and then continuously self-develop with new unlabelled samples. A weakly supervised scalable teacher forcing network for large-scale data streams (WeScatterNet) is presented in [40]. WeScatterNet features an ensemble framework with its base learners dynamically added and pruned via drift detection. WeScatterNet further extends the conventional fuzzily weighted generalized recursive least square (FWGRLS) algorithm by incorporating a novel regularization strategy that hinders important rules from accepting noisy pseudo labels. However, the key issue with these approaches is that they require externally controlled parameters to be predefined for pseudo-labelling and regularization. Such requirement brings subjective factors into the self-training process, and directly influences the structure and meta-parameters of the learned classification models. Without proper settings for these externally controlled parameters, the classification models may not be able to achieve satisfactory performance.

In this paper, a novel self-training EIS named semi-supervised self-organizing fuzzy inference system (S^3OFIS+) is proposed for semi-supervised learning from streaming data in indefinite delay environments [40]. The proposed S^3OFIS+ uses the simplified self-organizing fuzzy inference system (SOFIS+) [41] as its implementation basis. SOFIS+ is a recently introduced zero-order EIS capable of constructing precise classification boundaries from labelled data streams. By exploiting the pseudo labelling technique, S^3OFIS+ is capable of continuously self-calibrating more precise classification boundaries from unlabelled streaming data chunk-by-chunk in a non-iterative, exploratory manner after being primed. Utilizing the novel "$\mathcal{C}$ nearest prototypes ($\mathcal{C}$NP)" strategy (where

C represents the number of different classes in data) for pseudo labelling, S^{3}OFIS+ can assign pseudo labels to the most suitable candidates from each data chunk in a fully explainable manner and use them to self-expand its own knowledge base. The knowledge base of S^{3}OFIS+ is composed of human-interpretable prototypes, which represent the knowledge learned from data and are always meaningful. Thanks to its prototype-based nature, the system structure of S^{3}OFIS+ is highly transparent. Its reasoning and decision-making processes are based on the distances between data samples and the learned prototypes and, hence, are understandable and traceable to/by human. Very importantly, its semi-supervised learning scheme is free from externally controlled parameters but is entirely determined by the nature of data. To summarize, key contributions of this paper include:

1) a novel self-training EIS capable of continuously self-evolving from unlabelled streaming data "on the fly" with minimum human input;

2) a novel parameter-free pseudo-labelling strategy that can identity the most suitable unlabelled data samples for system updating in an objective and explainable manner.

The remainder of this paper is organized as follows. Section 2 summarizes the technical details of SOFIS+ as theoretical background. The self-training mechanism of S^{3}OFIS+ is described in Section 3 with the computational complexity analysis given by Section 4. Numerical examples are presented in Section 5 as the proof of concept, and this paper is concluded by Section 6.

2. Preliminaries: SOFIS+

In this section, technical details of SOFIS+ are recalled briefly to make this paper self-contained [41]. It is worth noting that SOFIS+ is a supervised EIS that learns its system structure and meta-parameters from labelled data. It is used as the implementation basis of the proposed S^{3}OFIS+.

2.1. Key Notations

First of all, let $\{\boldsymbol{x}\} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n, \dots\}$ ($\boldsymbol{x}_n = \left[x_{n,1}, x_{n,2}, \dots, x_{n,M}\right]^T \in \{\boldsymbol{x}\}$) be a particular data stream in a M dimensional real space, $\mathfrak{R}^M$, where the subscript n denotes the time instance at which the nth data sample $\boldsymbol{x}_n$, is observed. Samples of the data stream $\{\boldsymbol{x}\}$ continuously arrive in chunks, namely, $\{\boldsymbol{x}\}_k = \left\{\boldsymbol{x}_{k,1}, \boldsymbol{x}_{k,2}, \dots, \boldsymbol{x}_{k,L_k}\right\}$ ($k = 1,2, \dots, K, \dots$; L_k is the chunk size of $\{\boldsymbol{x}\}_k$). It is also assumed that $\{\boldsymbol{x}\}$ is composed of data samples of C different classes with $\{y\} = \{y_1, y_2, \dots, y_n, \dots\}$ being the corresponding labels, namely, y_n is the class label of $\boldsymbol{x}_n$ and there is $y_n \in \{1,2, \dots, C\}$. During the supervised learning process, the class labels of each labelled data chunk $\{\boldsymbol{x}\}_k$, denoted as $\{y\}_k$, are assumed to be available. Accordingly, $\{\boldsymbol{x}\}_k$ can be further divided into C non-overlapping subsets denoted as $\{\boldsymbol{x}\}_k^c = \left\{\boldsymbol{x}_{k,1}^c, \boldsymbol{x}_{k,2}^c, \dots, \boldsymbol{x}_{k,L_k^c}^c\right\}$, where $\{y\}_k^c = \left\{\overbrace{c, c, \dots, c}^{L_k^c}\right\}$ is the corresponding class labels of $\{\boldsymbol{x}\}_k^c$; L_k^c is the cardinality of $\{\boldsymbol{x}\}_k^c$ and $\{y\}_k^c$, and; there is $\sum_{j=1}^{C} L_k^j = L_k$. Note that different data chunks are not necessarily to be of the same size. For the purpose of generality, the most widely used Euclidean distance is employed as the default distance measure in this paper. However, other types of distance measure, i.e., Mahalanobis distance, cosine dissimilarity, can be considered as well depending on the nature of problems.

For readability purposes, key notations used in this paper and their definitions are summarized in Table 1.

Table 1. List of key notations and their definitions

Notation	Definition
$\{\boldsymbol{x}\}$	Data stream
$\{y\}$	Labels of $\{\boldsymbol{x}\}$
$\mathfrak{R}^M$	Real data space
M	Dimensionality of $\mathfrak{R}^M$
$\boldsymbol{x}_n$	The nth data sample
y_n	Label of $\boldsymbol{x}_n$
C	Number of classes
$\{\boldsymbol{x}\}_k$	The kth data chunk
$\{y\}_k$	Labels of $\{\boldsymbol{x}\}_k$
L_k	Chunk size of $\{\boldsymbol{x}\}_k$
$\{\boldsymbol{x}\}_k^c$	Data samples of the cth class within $\{\boldsymbol{x}\}_k$
$\{y\}_k^c$	Labels of $\{\boldsymbol{x}\}_k^c$

L_k^c	Cardinality of $\{\boldsymbol{x}\}_k^c$
$\boldsymbol{x}_{k,i}^c$	The ith data sample of $\{\boldsymbol{x}\}_k^c$
$\mathbf{R}^c$	The cth fuzzy rule
$\mathbf{P}^c$	Collection of prototypes of the cth class
P^c	Cardinality of $\mathbf{P}^c$
$\gamma_{k,g}^c$	Data-driven distance threshold derived from $\{\boldsymbol{x}\}_k^c$ at the gth level of granularity
$\boldsymbol{\mu}_{L_k^c}^c$	Arithmetic mean of $\{\boldsymbol{x}\}_k^c$
$X_{L_k^c}^c$	Arithmetic mean of $\{\|\boldsymbol{x}\|^2\}_k^c$
$\mathbf{p}_k^c$	Collection of prototypes of the cth class identified from $\{\boldsymbol{x}\}_k^c$
P_k^c	Cardinality of $\mathbf{p}_k^c$
$\boldsymbol{p}_j^c$	The jth prototype of $\mathbf{R}^c$
S_j^c	Number of data samples associated with $\boldsymbol{p}_j^c$

2.2. General Architecture

General architecture of SOFIS+ is depicted in Fig. 1.

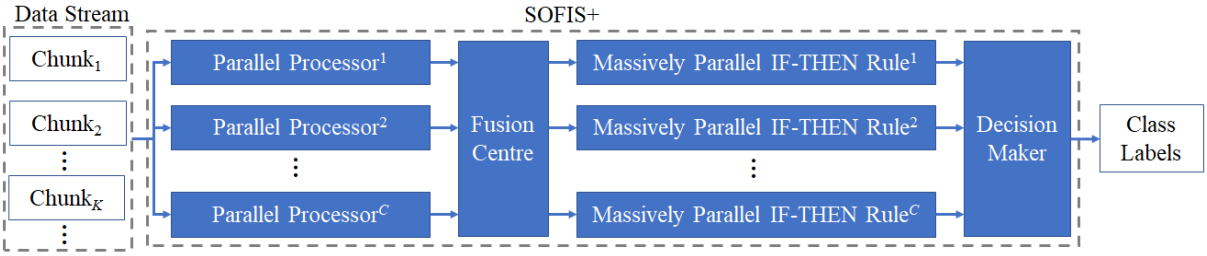


Fig. 1. General architecture of SOFIS+ [41]

It can be seen from Fig. 1 that SOFIS+ consists of C parallel processors, one fusion centre, C massively parallel IF-THEN fuzzy rules and one decision maker [41]. In this study, all the components of SOFIS+ are implemented on a single computation node. However, for large-scale data stream processing, parallel processors of SOFIS+ can be implemented on distributed nodes separately to facilitate computation.

During the learning stage, the parallel processors work simultaneously to extract knowledge from individual data chunks in the form of prototypes. The fusion centre then aggregates prototypes extracted from successive data chunks together to construct a compact knowledge base and build precise decision boundaries. The knowledge base is the key component of SOFIS+. It is composed of C massively parallel IF-THEN rules (one rule per class) formulated in the following form [10]:

$$\mathbf{R}^c: \quad \begin{array}{c} IF\ (\boldsymbol{x}\sim\boldsymbol{p}_1^c)\ OR\ (\boldsymbol{x}\sim\boldsymbol{p}_2^c)\ OR\ \ldots OR\ (\boldsymbol{x}\sim\boldsymbol{p}_{P^c}^c) \\ THEN\ (class\ c) \end{array} \tag{1}$$

where "$\sim$" denotes similarity; $\boldsymbol{p}_i^c \in \mathbf{P}^c$ is the ith prototype of $\mathbf{R}^c$, $i = 1,2,\ldots,P^c$; P^c is the number of prototypes identified from data samples of the cth class; $\mathbf{P}^c$ is the collection of prototypes of the cth class; $c = 1,2,\ldots,C$.

It is worth noting that SOFIS+ will discard historical data chunks to maintain high-level of computation- and memory- efficiency. In the next two subsections, the learning and decision-making protocols of SOFIS+ are presented.

2.3. Learning Protocol

The learning procedure of SOFIS+ is described as follows. By default, the externally controlled level of granularity is set as G.

Stage 1. Identifying prototypes

Given a new data chunk $\{\boldsymbol{x}\}_k$ with class labels $\{y\}_k$, SOFIS+ firstly divides $\{\boldsymbol{x}\}_k$ into C different subsets according to $\{y\}_k$, namely, $\{\boldsymbol{x}\}_k^1$, $\{\boldsymbol{x}\}_k^2$,…, $\{\boldsymbol{x}\}_k^C$, and passes the C subsets to the corresponding parallel processors (one processor per class).

After the cth processor has received $\{\boldsymbol{x}\}_k^c$, it estimates the data-driven distance threshold based on the mutual distances between samples of $\{\boldsymbol{x}\}_k^c$ and the level of granularity using Eqn. (2):

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

where $g = 1,2,\ldots,G$; $\gamma_{k,0}^{c} = 2\left(X_{L_k^c}^{c} - \left\|\boldsymbol{\mu}_{L_k^c}^{c}\right\|^2\right)$; $\boldsymbol{\mu}_{L_k^c}^{c}$ and $X_{L_k^c}^{c}$ are the respective arithmetic means of $\{\boldsymbol{x}\}_k^c$ and $\{\|\boldsymbol{x}\|^2\}_k^c$; $\|\boldsymbol{x}\|$ is the Euclidean norm of $\boldsymbol{x}$, namely, $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$; $N_{k,g}^{c}$ is the number of sample pairs in $\{\boldsymbol{x}\}_k^c$ between which the squared Euclidean distance is no greater than $\gamma_{k,g-1}^{c}$.

Then, the *c*th processor initializes the first prototype of the *c*th class with the first sample, $\boldsymbol{x}_{k,1}^{c}$ of $\{\boldsymbol{x}\}_k^c$:

$$P_k^c \leftarrow 1; \quad \boldsymbol{p}_1^c \leftarrow \boldsymbol{x}_{k,1}^c; \quad S_1^c \leftarrow 1; \quad \mathbf{p}_k^c \leftarrow \{\boldsymbol{p}_1^c\} \tag{3}$$

where $\mathbf{p}_k^c$ is the collection of prototypes of the *c*th class identified at the current learning cycle; $\boldsymbol{p}_j^c$ denotes the *j*th prototype identified from $\{\boldsymbol{x}\}_k^c$, $j = 1,2,\ldots,P_k^c$; P_k^c is the cardinality of $\mathbf{p}_k^c$; S_j^c is the support of $\boldsymbol{p}_j^c$, namely, the number of data samples associated with $\boldsymbol{p}_j^c$.

Next, each individual data sample remaining in $\{\boldsymbol{x}\}_k^c$ is examined by Condition 1 (Eqn. (4)), sequentially ($i = 2,3,\ldots,L_k^c$) [41]:

$$Condition\ 1: \quad \begin{array}{c} if\ \left(\min\limits_{\boldsymbol{p} \in \mathbf{p}_k^c}\left(\left\|\boldsymbol{p}-\boldsymbol{x}_{k,i}^c\right\|^2\right) > \bar{\gamma}_G^c\right) \\ then\ \left(\boldsymbol{x}_{k,i}^c\ becomes\ a\ new\ prototype\right) \end{array} \tag{4}$$

where $\bar{\gamma}_G^c = \frac{1}{\sum_{j=1}^{k} L_j^c} \sum_{i=1}^{k} L_i^c \gamma_{i,G}^c$. If $\boldsymbol{x}_{k,i}^c$ satisfies Condition 1, $\boldsymbol{x}_{k,i}^c$ represents an unfamiliar data pattern different from all existing prototypes within $\mathbf{p}_k^c$. In this case, $\boldsymbol{x}_{k,i}^c$ will be recognized as a new prototype:

$$P_k^c \leftarrow P_k^c + 1; \quad \boldsymbol{p}_{P_k^c}^c \leftarrow \boldsymbol{x}_{k,i}^c; \quad S_{P_k^c}^c \leftarrow 1; \quad \mathbf{p}_k^c \leftarrow \mathbf{p}_k^c \cup \left\{\boldsymbol{p}_{P_k^c}^c\right\} \tag{5}$$

Otherwise, $\boldsymbol{x}_{k,i}^c$ is used for updating the nearest prototype [10]:

$$\boldsymbol{p}_{n^*}^c \leftarrow \frac{S_{n^*}^c \boldsymbol{p}_{n^*}^c + \boldsymbol{x}_{k,i}^c}{S_{n^*}^c + 1}; \quad S_{n^*}^c \leftarrow S_{n^*}^c + 1 \tag{6}$$

where $\boldsymbol{p}_{n^*}^c = \underset{\boldsymbol{p} \in \mathbf{p}_k^c}{\operatorname{argmin}}\left(\left\|\boldsymbol{p}-\boldsymbol{x}_{k,i}^c\right\|^2\right)$.

After the prototype identification process has been completed, the newly identified prototypes, $\mathbf{p}_k^c$, $\mathbf{p}_k^c$, …, $\mathbf{p}_k^c$ are passed to the fusion centre, and SOFIS+ enters the next learning stage.

Stage 2. Self-calibrating classification boundaries

After the fusion centre has received $\mathbf{p}_k^c$, $\mathbf{p}_k^c$, …, $\mathbf{p}_k^c$, the knowledge base in the form of IF-THEN rules (Eqn. (1)) will be initialized ($\mathbf{P}^c \leftarrow \mathbf{p}_k^c$; $c = 1,2,\ldots,C$) if $\{\boldsymbol{x}\}_k$ is the very first data chunk (namely, $k = 1$). Otherwise (namely, $k > 1$), the fusion centre will firstly compare $\mathbf{p}_k^c$ with all prototypes of the same class identified from historical data chunks, denoted as $\mathbf{P}^c$ ($c = 1,2,\ldots,C$; P^c is the cardinality of $\mathbf{P}^c$). Condition 2 will be used to identify these more distinctive prototypes within $\mathbf{p}_k^c$ to join $\mathbf{P}^c$ ($j = 1,2,\ldots,P_k^c$) [41]:

$$Condition\ 2: \quad \begin{array}{c} if\ \left(\min\limits_{\boldsymbol{p} \in \mathbf{P}^c}\left(\left\|\boldsymbol{p}-\boldsymbol{p}_j^c\right\|^2\right) > \bar{\gamma}_G^c\right) \\ then\ \left(\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \left\{\boldsymbol{p}_j^c\right\}\right) \end{array} \tag{7}$$

Then, Condition 3 is utilized to identify candidate prototypes that are spatially close to the prototypes of other classes from the remaining members of $\mathbf{p}_k^c$ to join $\mathbf{P}^c$ [41]. These prototypes help SOFIS+ build more precise classification boundaries.

$$Condition\ 3: \quad \begin{array}{c} if\ \left(\min\limits_{\boldsymbol{p} \in \mathbf{p}_k^i}\left(\left\|\boldsymbol{p}-\boldsymbol{p}_j^c\right\|^2\right) \leq 2\bar{\gamma}_G^i\ \ \forall i \neq c\right) \\ then\ \left(\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \left\{\boldsymbol{p}_j^c\right\}\right) \end{array} \tag{8}$$

where $\boldsymbol{p}_j^c \in \mathbf{p}_k^c$; $c = 1,2,\ldots,C$.

After these newly identified prototypes have been aggregated into the knowledge base, the fuzzy rules are updated with the latest $\mathbf{P}^c$ ($c = 1,2,\ldots,C$). Then, SOFIS+ discards the current data chunk and goes back to Stage 1. A new

learning cycle begins with the next available data chunk (while setting $k \leftarrow k+1$) until no more new data samples available or being terminated by user.

Remark 1: The level of granularity, G is not a problem- or user- specific parameter that requires prior knowledge to be defined. Its value can be determined based on the preference of users, and the resulting data-driven threshold is guaranteed to be meaningful. In general, a greater value of G enables SOFIS+ (as well as S^{3}OFIS+) to identify more prototypes from data and build finer decision-boundaries. This usually leads to greater classification performance but lower computational efficiency. The recommended value for G is $G = 10$. It is also worth noting that one may estimate the most appropriate value for G directly from data using the elbow method described in [42]. This method only requires a regularization parameter to be determined by users. However, this estimation method is originally designed for offline applications, and it may not be able to make the best estimation in streaming environments because the underlying data patterns can change.

The learning procedure is also summarized by the following pseudo code [41].

Algorithm 1 SOFIS+ Identification

while (a new data chunk $\{\boldsymbol{x}\}_k$ with labels $\{y\}_k$ is available) **do**:
Stage 1. Identifying prototypes
  for $c = 1$ to C **do**:
    calculate $\gamma_{k,G}^c$ from $\{\boldsymbol{x}\}_k^c$ by Eqn. (2);
    initialize P_k^c, $\boldsymbol{p}_1^c$, S_1^c and $\mathbf{p}_k^c$ by Eqn. (3);
    for $i = 2$ to L_k^c **do**:
      if ($\boldsymbol{x}_{k,i}^c$ satisfies Condition 1) **then**
        add $\boldsymbol{x}_{k,i}^c$ as a new prototype by Eqn. (5);
      else
        update $\boldsymbol{p}_{n^*}^c$ and $S_{n^*}^c$ by Eqn. (6);
      end if
    end for
  end for
Stage 2. Self-calibrating classification boundaries
  for $c = 1$ to C **do**:
    if ($k = 1$) **then**
      $\mathbf{P}^c \leftarrow \mathbf{p}_k^c$;
      initialize $\mathbf{R}^c$ with $\mathbf{P}^c$;
    else
      expand $\mathbf{P}^c$ with $\mathbf{p}_k^c$ using Conditions 2 and 3;
      update $\mathbf{R}^c$ with $\mathbf{P}^c$;
    end if
  end for
end while

2.4. Decision-making Protocol

During the decision-making stage, for an unlabelled sample, $\boldsymbol{x}$, each IF-THEN rule will produce a confidence score based on the spatial similarity between $\boldsymbol{x}$ and its prototypes [10], [41]:

$$\lambda^c(\boldsymbol{x}) = \max_{\boldsymbol{p} \in \mathbf{P}^c} \left(e^{-\frac{\|\boldsymbol{x}-\boldsymbol{p}\|^2}{X-\|\boldsymbol{\mu}\|^2}} \right) \tag{9}$$

where $\boldsymbol{\mu}$ and X are the arithmetic means of $\{\boldsymbol{x}\}$ and $\{\|\boldsymbol{x}\|^2\}$, respectively.

The class label of $\boldsymbol{x}$ is determined by the decision maker based on the confidence scores produced by the C IF-THEN rules following the "winner takes all" principle [10], [41]:

$$label(\boldsymbol{x}) = c^*; \quad c^* = \underset{c=1,2,\ldots,C}{\operatorname{argmax}} \left(\lambda^c(\boldsymbol{x}) \right) \tag{10}$$

3. The Proposed S^{3}OFIS+

In this section, the semi-supervised learning protocol of the proposed S^{3}OFIS+ model is described in detail. As aforementioned, S^{3}OFIS+ uses SOFIS+ [41] as its implementation basis and exploits the idea of "pseudo labelling" [32] to perform self-training from unlabelled streaming data on a chunk-by-chunk basis with the aim of constructing a stronger prediction model. As aforementioned, this study considers the highly challenging

infinite delay problems [40], [43]. In such scenarios, only a reduced set of labelled data is available during the warming up stage [43]. This initially labelled data is necessary for defining the classification problems, i.e., numbers of classes, feature spaces. Note that in infinite delay problems, concept drifts may occur at any time regardless of the availability of labels. Typical characteristics of concept drifts can be $P(\boldsymbol{x})_t \neq P(\boldsymbol{x})_{t+1}$ and/or $P(y|\boldsymbol{x})_t \neq P(y|\boldsymbol{x})_{t+1}$ [37].

The self-training procedure of S^{3}OFIS+ consists of the following three stages. In Stage 0, S^{3}OFIS+ is firstly primed with the labelled data chunks $\{\boldsymbol{x}\}_k$ with the corresponding class labels $\{y\}_k$ in a supervised manner using Algorithm 1 ($k = 1,2, \ldots, K$). In Stage 1, S^{3}OFIS+ selects out a subset of unlabelled samples, $\{\hat{\boldsymbol{x}}\}_k$ from the current unlabelled chunk, $\{\boldsymbol{x}\}_k$ with the predicted class labels, $\{\hat{y}\}_k$ that S^{3}OFIS+ is highly confident with. In Stage 2, S^{3}OFIS+ uses $\{\hat{\boldsymbol{x}}\}_k$ and $\{\hat{y}\}_k$ to self-expand its knowledge base. After this, S^{3}OFIS+ goes back to Stage 1 and continues to self-learn from the next available unlabelled data chunk ($k = K+1, K+2, \ldots$). The self-training procedure is detailed as follows, and is also visualized in Fig. 2 in the form of flowchart for clarity.

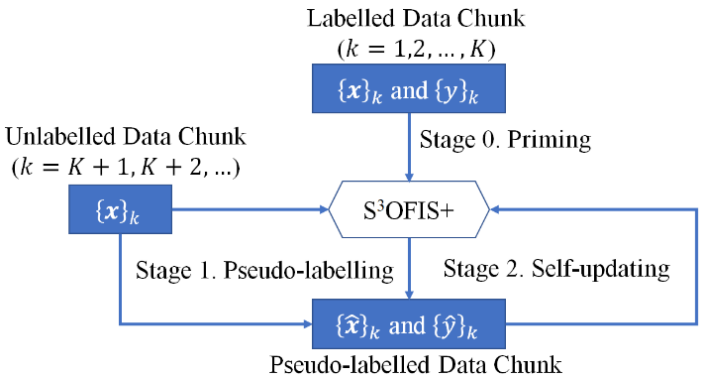


Fig. 2. Flowchart of the self-training procedure of S^{3}OFIS+

Stage 0. Priming classifier

In this stage, S^{3}OFIS+ is primed with labelled training samples on a chunk-by-chunk basis following the same algorithmic procedure presented in Section 2.3 (namely, Algorithm 1). Then, S^{3}OFIS+ enters the next stage to self-learn from unlabelled data streams.

Stage 1. Pseudo-labelling data

After being primed, S^{3}OFIS+ is ready for self-learning from unlabelled data chunks. As aforementioned, the main aim of Stage 1 is to assign class labels to unlabelled samples and identify these samples that can be used for expanding the knowledge base. S^{3}OFIS+ identifies such samples by using a novel, fully explainable strategy called "C nearest prototypes (*C*NP)", where C is the number of classes.

Conventional pseudo-labelling strategies [26], [27], [32] assign a pseudo label to a particular unlabelled sample based on the confidence scores produced by the classifier. For prototype-based self-training approaches [26], [27], [36], the confidence scores are usually calculated based on the distances between the unlabelled data sample and the nearest prototype of each class (one prototype per class; C prototypes in total), and an externally controlled threshold is usually required to help the classifier to identify the more confident decisions. However, such strategies suffer from two drawbacks: 1) the user-determined threshold can influence the pseudo-labelling outcomes and the classification performance of the resulting model; 2) the pseudo-labelling process is more sensitive to noisy and unstable because only one prototype per class is considered each time.

The proposed *C*NP strategy, on the other hand, overcomes the two drawbacks of conventional strategies by utilizing the C nearest prototypes selected from the entire knowledge base for pseudo-labelling. This brings two advantages to the proposed *C*NP strategy. Firstly, it is free from externally controlled threshold because the pseudo label of an unlabelled sample is determined by majority voting. Hence, the objectiveness is guaranteed. Secondly, *C*NP is more robust to noise than conventional pseudo-labelling strategies because it selects the C nearest prototypes from all existing prototypes for pseudo-labelling instead of selecting out just one nearest prototype from each class. Thus, the impact of noisy samples and outliers is minimized. One may also view the proposed *C*NP strategy as a variation of the well-known *k*-nearest neighbours (*k*NN) strategy [18] for determining the class labels. However, *C*NP differs from *k*NN in the following two aspects: 1) *C*NP is based on prototypes, which are

the most representative samples in the data space, rather than raw data samples; and 2) the number of nearest prototypes considered by *C*NP is determined by data, not by users.

Once a new unlabelled data chunk is available, $\{\boldsymbol{x}\}_k$ (namely, $\{y\}_k$ is not captured with $\{\boldsymbol{x}\}_k$), the current self-training cycle starts. For each unlabelled sample, $\boldsymbol{x}_{k,i} \in \{\boldsymbol{x}\}_k$, S^{3}OFIS+ will firstly identify C nearest prototypes from the knowledge base based on their spatial distances to $\boldsymbol{x}_{k,i}$, denoted as $\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \ldots, \boldsymbol{p}_C^* \in \mathbf{P}^1 \cup \mathbf{P}^2 \cup \ldots \cup \mathbf{P}^C$.

$\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \ldots, \boldsymbol{p}_C^*$ will vote together to determine the class label, $\hat{y}_{k,i}$ of $\boldsymbol{x}_{k,i}$ as follows:

$$vote\big(\hat{y}_{k,i} = c\big) = \sum_{j=1}^{C} \mathbb{I}\big(y_j^* = c\big) \tag{11}$$

where y_j^* is the class label of $\boldsymbol{p}_j^*$ and, $\mathbb{I}\big(y_j^* = c\big) = \begin{cases} 1, & if\ y_j^* = c \\ 0, & if\ y_j^* \neq c \end{cases}$.

The class label of $\boldsymbol{x}_{k,i}$ is determined as the class that receives the most votes, namely,

$$vote\big(\hat{y}_{k,i} = c^*\big) > \max_{\substack{j=1,2,\ldots,C; \\ j \neq c^*}} \Big(vote\big(\hat{y}_{k,i} = j\big)\Big) \tag{12}$$

However, $\boldsymbol{x}_{k,i}$ will be recognized as a tricky sample without being labelled if there are two or more classes winning together based on Eqn. (12). Then, Condition 4 is examined to see whether this prediction is a confident decision or not:

$$Condition\ 4\text{:} \quad \begin{matrix} if\ \Big(vote\big(\hat{y}_{k,i} = c^*\big) \geq \left\lfloor \frac{C}{2} \right\rfloor + 1\Big) \\ then\ \big(\hat{y}_{k,i}\ is\ a\ confident\ decision\big) \end{matrix} \tag{13}$$

where "$\lfloor \cdot \rfloor$" denotes the round-down operation.

If Condition 4 is satisfied, it suggests that more than half of the C nearest prototypes share the same class label, and it is highly likely that $\boldsymbol{x}_{k,i}$ also belongs to this class. In this case, $\boldsymbol{x}_{k,i}$ and $\hat{y}_{k,i}$ will be used by S^{3}OFIS+ to self-expand its knowledge base, namely, $\{\widehat{\boldsymbol{x}}\}_k \leftarrow \{\widehat{\boldsymbol{x}}\}_k \cup \{\boldsymbol{x}_{k,i}\}$ and $\{\hat{y}\}_k \leftarrow \{\hat{y}\}_k \cup \{\hat{y}_{k,i}\}$.

Otherwise, the votes by the C nearest prototypes are highly diversified. In this case, $\boldsymbol{x}_{k,i}$ could be a challenging sample and there is a chance that the predicted class label, $\hat{y}_{k,i}$ is wrong. As a result, $\boldsymbol{x}_{k,i}$ and $\hat{y}_{k,i}$ will not be used for system self-updating to avoid introducing potential pseudo-labelling errors to the system.

After S^{3}OFIS+ have checked every unlabelled sample in $\{\boldsymbol{x}\}_k$ and selected out a subset, $\{\widehat{\boldsymbol{x}}\}_k$ from $\{\boldsymbol{x}\}_k$ with pseudo labels $\{\hat{y}\}_k$ that it is confident with, the self-training cycle enters the second stage. The rest of $\{\boldsymbol{x}\}_k$ is discarded for memory efficiency.

Remark 2: The main purpose of the *C*NP strategy is to identify these confidently pseudo-labelled samples for S^{3}OFIS+ to self-improve its knowledge base and self-calibrate finer classification boundaries gradually. Although these more challenging/tricky samples may contain important information about new data patterns and can potentially be useful for sharpening classification boundaries, S^{3}OFIS+ avoid utilizing them in the absence of ground truth and/or additional assistance (e.g., human expertise) because such samples are more likely to introduce potential pseudo-labelling errors to the system, leading to poor classification precision.

To better illustrate the core idea of the proposed *C*NP strategy, a simple example demonstrating the pseudo labelling process of S^{3}OFIS+ is presented in Fig. 3. It can be seen from the left part of Fig. 3 that in this data space, there are a total of 16 prototypes of four different classes (four prototypes per class) represented by blue, green, red and yellow dots "●", respectively. In addition, there are four unlabelled samples represented by white dots "●". For every unlabelled sample, four nearest prototypes are identified ($C = 4$). The nearest prototypes surrounding each unlabelled samples are circulated by grey dash lines in the middle part of Fig. 3, and the pseudo-labelling results are given by the right part of Fig. 3. One can see that the unlabelled sample 1 is labelled as the blue class because three of its nearest prototypes belong to this class. The unlabelled samples 2 and 4 are labelled as the yellow class as the majority of their nearest prototypes are of the yellow class. However, the sample 2 will not be used for expanding the knowledge base because Condition 4 is not satisfied. The unlabelled sample 4 is not labelled because the blue and green classes receive the same amount of votes.

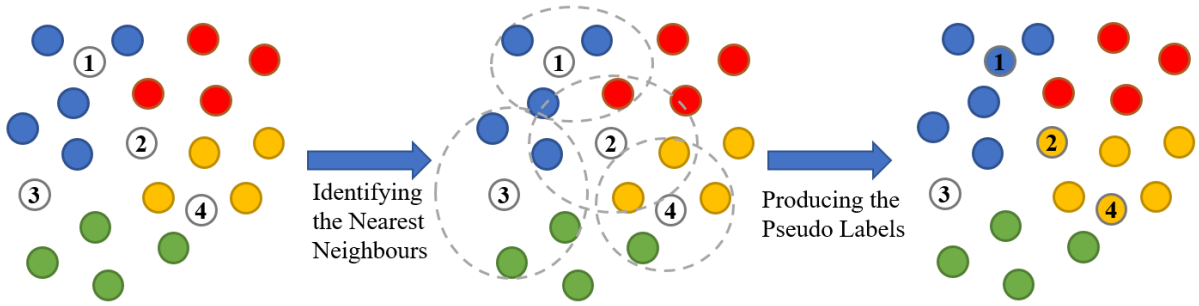


Fig. 3. Illustration of pseudo labelling using *C*NP principle

Stage 2. Self-updating classifier

In this stage, S^{3}OFIS+ performs self-training with $\{\hat{\boldsymbol{x}}\}_k$ and $\{\hat{y}\}_k$ to expand its knowledge base following the same procedure of Algorithm 1 in a supervised manner. Then, the S^{3}OFIS+ self-training process iterates, going back to Stage 1 and starting the next cycle given a new data chunk (while setting $k \leftarrow k+1$).

Remark 3: The knowledge base of S^{3}OFIS+ contains only these highly informative prototypes identified from data streams. During each self-training cycle, the newly identified prototypes will be updated by other pseudo-labelled samples within the same chunk such that the concept drifts in data streams are handled locally. After this, only these new prototypes that capture the latest changes of data patterns and/or contribute to building more precise classification boundaries are selected to join the knowledge base. In this way, S^{3}OFIS+ maintains a concise knowledge base while self-adapting to drifts in the data streams agilely.

The self-training procedure is summarized as follows.

Algorithm 2 S^{3}OFIS+ Self-training

Stage 0. Priming classifier
for $k = 1$ to K **do**:
  train the model with $\{\boldsymbol{x}\}_k$ and $\{y\}_k$ using ***Algorithm 1***;
end for
while (a new unlabelled data chunk $\{\boldsymbol{x}\}_k$ is available) **do**:
Stage 1. Pseudo-labelling data
  for $i = 1$ to L_k **do:**
    identify the nearest prototypes, $\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, \ldots, \boldsymbol{p}_C^*$ to $\boldsymbol{x}_{k,i}$;
    determine $\hat{y}_{k,i}$ through voting by Eqns. (11) and (12);
    if ($\hat{y}_{k,i}$ satisfies Condition 4) **then**
      $\{\hat{\boldsymbol{x}}\}_k \leftarrow \{\hat{\boldsymbol{x}}\}_k \cup \{\boldsymbol{x}_{k,i}\}$;
      $\{\hat{y}\}_k \leftarrow \{\hat{y}\}_k \cup \{\hat{y}_{k,i}\}$;
    end if
  end for
Stage 2. Self-updating classifier
  train the model with $\{\hat{\boldsymbol{x}}\}_k$ and $\{\hat{y}\}_k$ using ***Algorithm 1***;
end while

Remark 4: The unique supervised learning protocol (namely, Algorithm 1) provides S^{3}OFIS+ strong resistance to pseudo-labelling errors/noises by its nature. When aggregating newly identified prototypes from the current data chunk into the knowledge base, existing prototypes identified from historical data chunks remain intact to prevent error-propagation. In such way, pseudo-labelling errors at each self-training cycle are confined locally and distortions on the classification boundaries caused by these errors are minimized.

Remark 5: This study mainly considers infinite delay problems. Nevertheless, S^{3}OFIS+ is also suitable for the applications where labelled and unlabelled data arrived simultaneously at random. In such cases, S^{3}OFIS+ will firstly learn from the labelled data within the current data chunk in a supervised manner (namely, Algorithm 1) and then continue to perform self-training on the remaining unlabelled samples following the algorithmic procedure described in this section (namely, Stages 1 and 2 in Algorithm 2).

4. Computational Complexity Analysis

Analysis on the computational complexity of the proposed S^{3}OFIS+ is provided in this section.

4.1. Learning Procedure

Supervised learning (Stage 0): S^{3}OFIS+ learns from labelled data chunks to prime its knowledge base using ***Algorithm 1***. In the first stage of ***Algorithm 1***, the computational complexity of calculating data-driven thresholds, $\gamma_{k,G}^1$, $\gamma_{k,G}^2$,..., $\gamma_{k,G}^C$ based on the mutual distances between data samples of the current data chunk $\{\boldsymbol{x}\}_k$ is $O(M\sum_{c=1}^{C}(L_k^c)^2)$. The complexity of identifying prototypes, $\mathbf{p}_k^1, \mathbf{p}_k^2, \ldots, \mathbf{p}_k^C$ by Condition 1 is $O(\sum_{c=1}^{C} L_k^c)$, and that of updating prototypes is $O(M\sum_{c=1}^{C} P_k^c)$. Next, if $\{\boldsymbol{x}\}_k$ is not the very first data chunk, these newly identified prototypes will be aggregated into the knowledge base (namely, $\mathbf{P}^1, \mathbf{P}^2, \ldots, \mathbf{P}^C$) learned from historical data chunks. The computational complexity of using Conditions 2 and 3 to select candidate prototypes are $O(M\sum_{c=1}^{C} P_k^c P^c)$ and $O\left(M\sum_{l=1}^{C-1}\sum_{j=l+1}^{C} P_k^l P_k^j\right)$, respectively. Assume that there are K labelled data chunks available, the computational complexity of the classifier priming process is $O(M\sum_{k=1}^{K}\sum_{c=1}^{C}((L_k^c)^2 + P_k^c P^c))$.

Self-training (Stages 1 and 2): After the knowledge base has been primed with labelled data, S^{3}OFIS+ enters the self-training stage. Given an unlabelled data chunk $\{\boldsymbol{x}\}_k$, S^{3}OFIS+ firstly identifies the C nearest prototypes to each unlabelled sample for pseudo-labelling, and the computational complexity of this process is $O(ML_k\sum_{c=1}^{C} P^c)$. The computational complexity of identifying pseudo labels with high confidence by Condition 4 is negligible. Then, S^{3}OFIS+ updates its knowledge base with the pseudo-labelled data $\{\hat{\boldsymbol{x}}\}_k$ and $\{\hat{y}\}_k$ using ***Algorithm 1***, and the computational complexity of this is $O\left(M\sum_{c=1}^{C}\left(\left(\hat{L}_k^c\right)^2 + \hat{P}_k^c P^c\right)\right)$, where $\hat{L}_k^c$ and $\hat{P}_k^c$ are the respective numbers of pseudo-labelled samples and newly identified prototypes of the cth class. Assume that there are U unlabelled data chunks available, the computational complexity of the classifier self-training process is $O\left(M\sum_{k=1}^{U}\sum_{c=1}^{C}\left(\left(\hat{L}_k^c\right)^2 + P^c\left(\hat{P}_k^c + L_k\right)\right)\right)$.

Hence, it can be concluded from the above analysis that the overall computational complexity of the classifier learning process (including both supervised learning and self-training) is $O\left(M\left(\sum_{k=1}^{K}\sum_{c=1}^{C}((L_k^c)^2 + P_k^c P^c) + \sum_{k=1}^{U}\sum_{c=1}^{C}\left(\left(\hat{L}_k^c\right)^2 + P^c\left(\hat{P}_k^c + L_k\right)\right)\right)\right)$

4.2. Decision-making Procedure

During the decision-making process, the computational complexity of calculating the distances between a particular testing sample and prototypes in the knowledge base, namely, $\mathbf{P}^1, \mathbf{P}^2, \ldots, \mathbf{P}^C$ is $O(M\sum_{c=1}^{C} P^c)$. For L testing samples, the overall computational complexity to determine their class labels is $O(ML\sum_{c=1}^{C} P^c)$.

5. Experimental Investigation

5.1. Configuration

To evaluate the performance of the proposed S^{3}OFIS+ system, numerical examples based on a wide range of benchmark problems from UCI Machine Learning Repository[1], Keel Dataset Repository[2] and Scikit-Multiflow[3] are presented. As aforementioned, all numerical examples in this paper are performed in infinite delay scenarios [40], [43]. The datasets used for experimental studies are as follows.

1) Australian (AU) dataset;
2) Banknote authentication (BA) dataset;
3) Gesture phase segmentation (GP) dataset;
4) Image segmentation (IS) dataset;
5) Multiple feature (MF) dataset;
6) Magic gamma telescope (MG) dataset;
7) Occupancy detection (OD) dataset
8) Optical recognition of handwritten digits (OR) dataset;
9) Page-blocks (PB) dataset;
10) Pen-based recognition of handwritten digits (PR) dataset;
11) Shill bidding (SB) dataset ;

[1] Available at: https://archive.ics.uci.edu/ml/index.php
[2] Available at: https://sci2s.ugr.es/keel/index.php
[3] Available at: https://scikit-multiflow.github.io/

12) Segment (SE) dataset;
13) Semeion handwritten digit (SH) dataset;
14) Texture (TE) dataset;
15) Wilt (WI) dataset;
16) SEA dataset[4], and;
17) Hyperplane (HYP) dataset[5].

In addition, the popular MNIST image set[6] for handwritten digit recognition and its two nonstationary synthetic variations, namely, permuted MNIST (PMNIST) and rotated MNIST (RMNIST)[7] are also used for experimental studies. Key information of the 20 datasets is summarized in Table 2.

Table 2. Key information of benchmark datasets for performance evaluation

Dataset	#Classes	#Samples	#Attributes	Characteristics	Class Proportion
AU	2	690	14	stationary	55.5;44.5
BA	2	1372	4	stationary	55.5;44.5
GP	5	9900	18	stationary	28.0;21.2;29.8;10.1;11.0
IS	7	2310	19	stationary	14.3;14.3;14.3;14.3;14.3;14.3;14.3
MF	10	2000	649	stationary	10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0
MG	2	19020	10	stationary	64.8;35.2
OD	2	20560	5	stationary	76.9;23.1
OR	10	5620	64	stationary	9.9;10.2;9.9;10.2;10.1;9.9;9.9;10.1;9.9;10.0
PB	5	5472	10	stationary	89.8;6.0;0.5;1.6;2.1
PR	10	10992	16	stationary	10.4;10.4;10.4;9.6;10.4;9.6;9.6;10.4;9.6;9.6
SB	2	6321	9	stationary	10.7;89.3
SE	7	2310	19	stationary	14.3;14.3;14.3;14.3;14.3;14.3;14.3
SH	10	1593	256	stationary	10.1;10.2;10.0;10.0;10.1;10.0;10.1;9.9;9.7;9.9
TE	11	5500	40	stationary	9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1
WI	2	4839	5	stationary	5.4;94.6
SEA	2	120000	3	nonstationary	24.5;75.5
HYP	2	25000	4	nonstationary	50.0;50.0
MNIST	10	70000	28×28	stationary	9.9;11.3;10.0;10.2;9.7;9.0;9.8;10.4;9.8;9.9
PMNIST	10	70000	784	nonstationary	9.9;11.3;10.0;10.2;9.7;9.0;9.8;10.4;9.8;9.9
RMNIST	10	62000	784	nonstationary	9.9;11.2;10.0;10.2;9.8;9.0;9.8;10.4;9.8;10.0

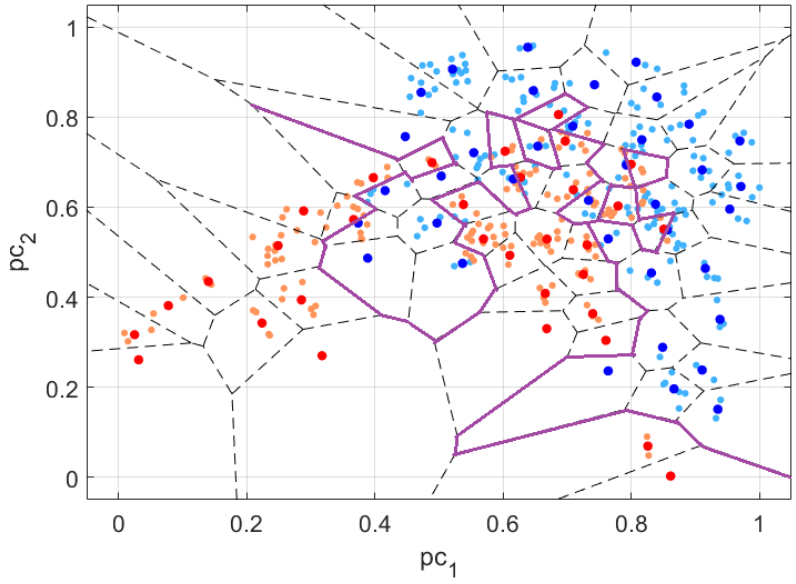
The numerical examples presented in this paper are conducted on a laptop with dual core i7 processer $3.60GHz \times 2$ and $16GB$ RAM. The algorithms are developed on MATLAB2020b platform. The reported results are obtained as the average of 10 Monte Carlo experiments unless specifically declared otherwise. It is worth noting that as S^{3}OFIS+ is designed to learn from streaming data on a chunk-by-chunk basis, during the numerical experiments, only one data chunk is presented to S^{3}OFIS+ each time to simulate the online learning environment. The MATLAB code of S^{3}OFIS+ is publicly available[8].

5.2. Visual Demonstration

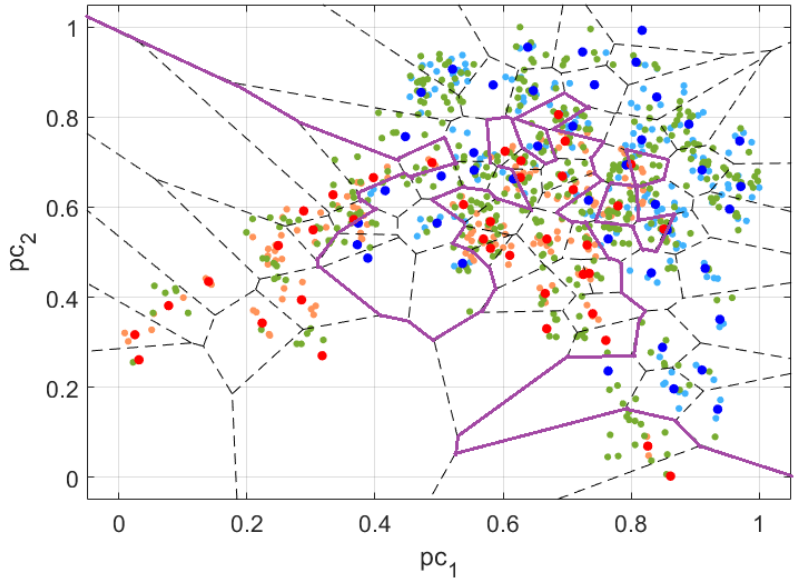
In this subsection, a numerical example is presented to visualize the process of S^{3}OFIS+ constructing a stronger model with higher precision and building finer classification boundaries from both labelled and unlabelled data on a chunk-by-chunk basis. In this example, the BA dataset is used thanks to its smaller scale and simpler structure. For visual clarity, principle component analysis is applied to reduce the dimensionality of data from four to two. The BA dataset is divided into four chunks evenly with the first chunk used as the labelled chunk and the remaining three as the unlabelled ones. During the experiment, S^{3}OFIS+ is trained by the labelled chunk to prime the knowledge base and then self-trained with the three unlabelled chunks one-by-one to further self-improve its classification boundaries without human supervision. The level of granularity is set as $G = 4$ in this experiment. The classification boundaries constructed by S^{3}OFIS+ from the first labelled chunk are depicted in Fig. 4(a), and the updated classification boundaries after self-training with the first, second and third unlabelled chunks are depicted in Fig. 4(b)-(d), respectively. In Fig. 4, the light blue and pink points "·" represent labelled data samples of the two classes; the green points "·" are the data samples without labels; the blue and red dots "●" are the

[4] Generated by SEAGenerator API from Scikit-Multiflow
[5] Generated by HyperplaneGenerator API from Scikit-Multiflow
[6] Available at: http://yann.lecun.com/exdb/mnist/
[7] Available at: https://nlp.stanford.edu/projects/mer/
[8] Available at: https://github.com/Gu-X/Semi-Supervised-Self-Organizing-Fuzzy-Inference-System

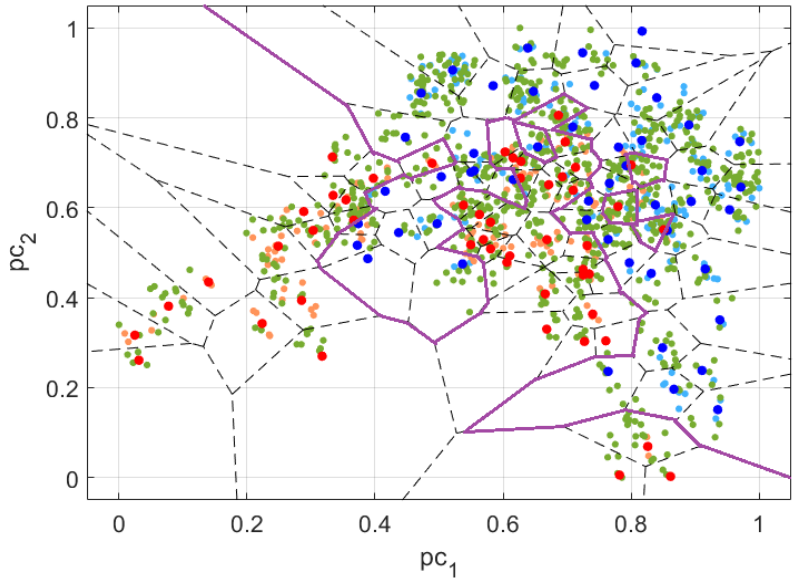
respective prototypes of the two classes; the dash lines "--" are the boundaries of Voronoi tessellations formed around prototypes; the purple lines "-" are the classification boundaries. It can be observed from this example that after being primed with the labelled data chunk, S^{3}OFIS+ is able to continuously self-improve its decision-boundaries with the successive unlabelled data chunks, showing the effectiveness of the proposed self-training mechanism.



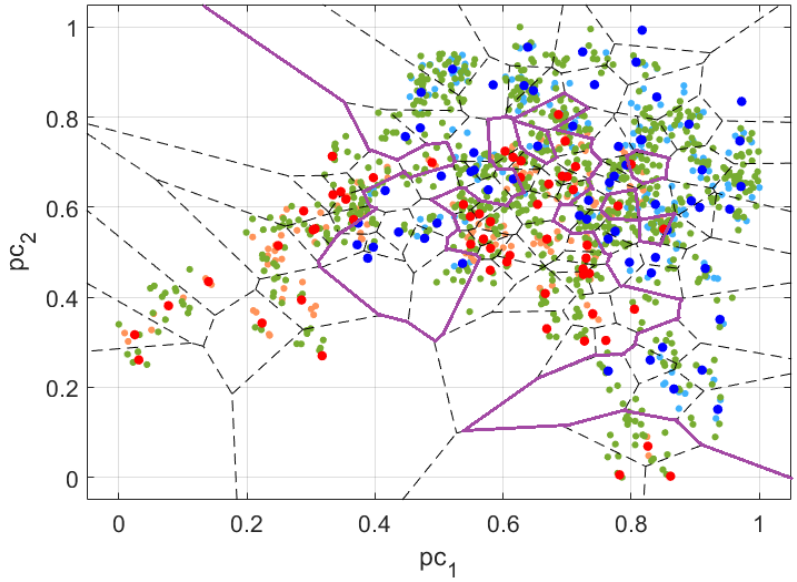
(a) Classification boundaries constructed from the labelled chunk



(b) Classification boundaries constructed with the first unlabelled chunk



(c) Classification boundaries constructed with the second unlabelled chunk



(d) Classification boundaries constructed with the third unlabelled chunk

Fig. 4. Illustration of constructing more precise classification boundaries with labelled and unlabelled data chunks

5.3. Sensitivity Analysis

In this subsection, sensitivity analysis is conducted to investigate the influence of 1) level of granularity, G and 2) chunk size, L on the classification performance of S^{3}OFIS+.

Firstly, the influence of G on system performance is investigated. The following five benchmark datasets, OD, OR, PB, PR and WI are employed. During the experiments, 10% of data is randomly selected to form the labelled training set and the remaining data is used to build the unlabelled one. The value of G varies from 3 to 11 and L is set as $L = 1000$. The classification performance of S^{3}OFIS+ in terms of classification accuracy rate (Acc) on the unlabelled sets after learning from both labelled and unlabelled data are reported in Table 3. To understand the influence of G on the computational efficiency of the system, the computation time consumptions (t_{exe}, in seconds) with different experimental settings are also reported in the same table. It can be seen from Table 3 that

the level of granularity has a direct impact on both the classification accuracy and computational efficiency of the system. In general, a higher level of granularity can boost the classification accuracy of S^{3}OFIS+, but it also increases the computational complexity because more prototypes are identified from data, leading to higher computation time consumption. In the rest of this paper, $G = 10$ is used as a trade-off between accuracy and efficiency.

Table 3. Investigation of influence of G on system performance ($L = 1000$)

G	Measure	Dataset				
		OD	OR	PB	PR	WI
3	Acc	0.9152	0.9649	0.6038	0.9439	0.6810
	t_{exe}	1.2808	0.5145	0.3550	1.0120	0.2309
4	Acc	0.9173	0.9670	0.6536	0.9599	0.7568
	t_{exe}	1.3195	0.5863	0.3848	0.8609	0.2674
5	Acc	0.9198	0.9678	0.7444	0.9717	0.8237
	t_{exe}	1.4560	0.6549	0.4366	0.9535	0.2894
6	Acc	0.9559	0.9697	0.8076	0.9762	0.8886
	t_{exe}	1.5049	0.7190	0.4604	1.1410	0.3231
7	Acc	0.9724	0.9697	0.8439	0.9792	0.9438
	t_{exe}	1.6036	0.7846	0.4989	1.2576	0.3430
8	Acc	0.9814	0.9710	0.8727	0.9805	0.9631
	t_{exe}	1.7909	0.8414	0.5297	1.4924	0.3655
9	Acc	0.9849	0.9709	0.9087	0.9806	0.9655
	t_{exe}	1.9465	0.8582	0.5608	1.5947	0.4023
10	Acc	0.9863	0.9712	0.9250	0.9810	0.9646
	t_{exe}	2.2997	0.8908	0.6057	1.7428	0.4282
11	Acc	0.9867	0.9713	0.9315	0.9810	0.9642
	t_{exe}	2.7262	0.9181	0.6598	1.7895	0.4578

Next, the influence of L on the classification performance of S^{3}OFIS+ is investigated. In this example, the same experimental protocol as used in the previous example is utilized. During the experiments, the value of L varies from 200 to 2000 and G is set as $G = 10$. The classification performances of S^{3}OFIS+ in terms of Acc and t_{exe} are tabulated in Table 4. One can see form Table 4 that generally, a greater L helps S^{3}OFIS+ to achieve higher classification accuracy and, at the same time, reduces its computational efficiency. The main reason for this is because data chunks with a larger size enable S^{3}OFIS+ to have a better understanding of the underlying data patterns, but it would require more time for S^{3}OFIS+ to process each individual data chunk and fuse the learned knowledge together. In the numerical examples presented in the rest of Section 5, $L = 1000$ is used unless specifically declared otherwise.

Table 4. Investigation of influence of G on system performance ($G = 10$)

L	Measure	Dataset				
		OD	OR	PB	PR	WI
200	Acc	0.9862	0.9660	0.9345	0.9773	0.9594
	t_{exe}	2.1422	0.7811	0.5720	1.3193	0.4151
400	Acc	0.9863	0.9687	0.9336	0.9812	0.9601
	t_{exe}	2.2177	0.8280	0.5365	1.6395	0.4116
600	Acc	0.9864	0.9711	0.9307	0.9814	0.9637
	t_{exe}	2.3384	0.8802	0.5540	1.7462	0.4054
800	Acc	0.9864	0.9719	0.9258	0.9808	0.9644
	t_{exe}	2.3571	0.9065	0.5678	1.7501	0.4275
1000	Acc	0.9863	0.9712	0.9250	0.9810	0.9646
	t_{exe}	2.2997	0.8908	0.6057	1.7428	0.4282
1500	Acc	0.9861	0.9698	0.9248	0.9809	0.9650
	t_{exe}	2.4737	0.9119	0.6364	1.7849	0.4806
2000	Acc	0.9862	0.9696	0.9236	0.9809	0.9652
	t_{exe}	2.6583	0.8655	0.6812	1.7959	0.5141

5.4. Ablation Analysis

Ablation analysis is performed in this subsection to test the effectiveness of the proposed self-training mechanism. In this example, the classification performances of S^{3}OFIS+ and SOFIS+ are compared on 14 datasets, which include AU, GP, IS, MG, MF, OD, OR, PB, PR, SB, SE, SH, TE and WI. During the experiments, 10% of data is randomly selected to build the labelled training set and the rest is used to form the unlabelled training set. For visual clarity, the classification error rates on the unlabelled data obtained by S^{3}OFIS+ and SOFIS are reported in Table 5, and the performance improvements in percentage are also reported following the common practice [28]. In addition, the same experiments are repeated with 20%, 30% and 40% of data used as labelled training set. The classification error rates of S^{3}OFIS+ and SOFIS under the three splitting ratios are reported in the same table. The performance improvements in percentage are given in Table 5 as well. For visual clarity, the average classification error rates of S^{3}OFIS+ and SOFIS+ under different split ratios are also depicted in Fig. 5.

It can be seen from Table 5 that with 10%, 20%, 30% and 40% of labelled training data, S^{3}OFIS+ is able to reduce its average classification error rates by 5.40%, 6.14%, 5.13% and 3.10% across the 14 benchmark classification problems. The numerical results demonstrate the efficacy of the proposed semi-supervised learning mechanism enabling S^{3}OFIS+ utilize unlabelled data to construct a more precise prediction model without human supervision.

Table 5. Ablation analysis results in terms of classification error rates

Dataset	% Labelled	Algorithm		Δ %	% Labelled	Algorithm		Δ %
		S^{3}OFIS+	SOFIS+			S^{3}OFIS+	SOFIS+	
AU	10%	0.3981	0.4225	+6.13%	20%	0.3808	0.4096	+7.56%
GP		0.2850	0.2761	-3.12%		0.2072	0.2019	-2.56%
IS		0.1271	0.1338	+5.27%		0.0812	0.0838	+3.20%
MF		0.1079	0.1104	+2.32%		0.0856	0.0846	-1.17%
MG		0.2317	0.2511	+8.37%		0.2262	0.2410	+6.54%
OD		0.0137	0.0164	+19.71%		0.0145	0.0150	+3.45%
OR		0.0288	0.0350	+21.53%		0.0212	0.0241	+13.68%
PB		0.0750	0.0652	-13.07%		0.0649	0.0615	-5.24%
PR		0.0190	0.0214	+12.63%		0.0128	0.0141	+10.16%
SB		0.0126	0.0121	-3.97%		0.0048	0.0062	+29.17%
SE		0.1369	0.1407	+2.78%		0.0900	0.0936	+4.00%
SH		0.2090	0.2144	+2.58%		0.1575	0.1625	+3.17%
TE		0.0435	0.0467	+7.36%		0.0279	0.0296	+6.09%
WI		0.0354	0.0379	+7.06%		0.0290	0.0313	+7.93%
Average		**0.1231**	**0.1274**	**+5.40%**		**0.1003**	**0.1042**	**+6.14%**
AU	30%	0.3870	0.4072	+5.22%	40%	0.3848	0.3986	+3.59%
GP		0.1684	0.1611	-4.33%		0.1366	0.1324	-3.07%
IS		0.0661	0.0698	+5.60%		0.0556	0.0562	+1.08%
MG		0.2221	0.2366	+6.53%		0.2203	0.2311	+4.90%
MF		0.0725	0.0723	-0.28%		0.0649	0.0637	-1.85%
OD		0.0144	0.0153	+6.25%		0.0150	0.0155	+3.33%
OR		0.0180	0.0191	+6.11%		0.0167	0.0170	+1.80%
PB		0.0609	0.0588	-3.45%		0.0581	0.0574	-1.20%
PR		0.0103	0.0107	+3.88%		0.0095	0.0101	+6.32%
SB		0.0034	0.0044	+29.41%		0.0032	0.0037	+12.50%
SE		0.0738	0.0753	+2.03%		0.0627	0.0657	+4.78%
SH		0.1300	0.1330	+2.31%		0.1238	0.1238	+0.00%
TE		0.0214	0.0228	+6.54%		0.0175	0.0180	+2.86%
WI		0.0249	0.0264	+6.02%		0.0239	0.0259	+8.37%
Average		**0.0909**	**0.0938**	**+5.13%**		**0.0852**	**0.0871**	**+3.10%**

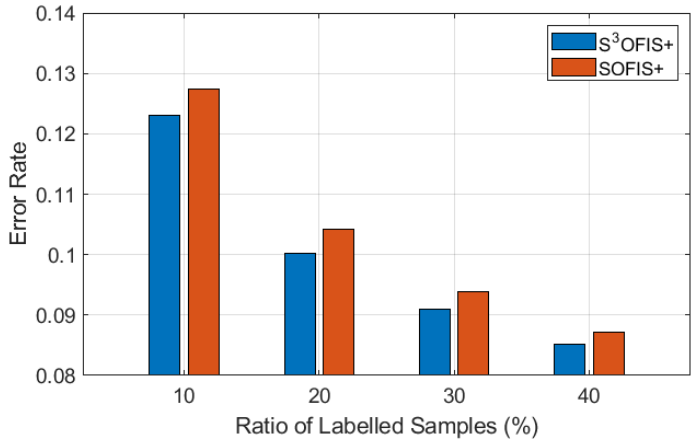


Fig. 5. Average classification error rate comparison between S^{3}OFIS+ and SOFIS+

5.5. Performance Comparison

In this subsection, the performance of S^{3}OFIS+ is compared with a wide variety of semi-supervised and supervised machine learning algorithms on the aforementioned classification problems as a benchmark comparison.

Firstly, the classification accuracy and computational efficiency of S^{3}OFIS+ is compared with the following semi-supervised learning algorithms on the 14 numerical benchmark datasets (namely, AU, GP, IS, MG, MF, OD, OR, PB, PR, SB, SE, SH, TE and WI) used in the previous example:

1) LGC classifier [29];
2) LapSVM classifier [30];
3) Anchor graph regularization with kernel weights (Anchor-K) classifier [31];
4) Anchor graph regularization with local anchor embedding weights (Anchor-L) classifier [31]
5) Efficient anchor graph regularization (EAnchor) classifier [44];
6) SeRBIA classifier [26], and;
7) STHP classifier [27].

Note that LGC, LapSVM, Anchor-K, Anchor-L and EAnchor are among the most popular transductive semi-supervised learning algorithms to learn from static data in offline application scenarios. Thus, for LGC, LapSVM, Anchor-K, Anchor-L and EAnchor, the numerical experiments are performed in offline scenarios, namely, all the labelled and unlabelled data samples are available to the classifiers. SeRBIA and STHP are recently proposed self-training algorithms to learn from data on a chunk-by-chunk basis and can be implemented for both offline and online application sceneries, same as S^{3}OFIS+. Since SeRBIA and STHP both aim to learn a set of prototypes from both labelled and unlabelled data for classification via pseudo labelling, the performance comparison between SeRBIA, STHP and S^{3}OFIS+ is essentially equivalent to the comparison between different pseudo-labelling mechanisms employed. The numerical experiments with SeRBIA and STHP, are performed in online scenarios following the same experimental protocols as S^{3}OFIS+.

During the experiments, LGC used the kNN graph with $k = 5$ and α is set as $\alpha = 0.99$ [29]. LapSVM uses the "one versus all" strategy for multi-class classification tasks and used the radial basis function kernel. As the performance of LapSVM is very sensitive to the predefined parameters, three different experimental settings are considered here, namely, *i)* $\sigma = 10$, $\mu_I = 1$, $\mu_A = 10^{-6}$, $k = 15$ (as suggested by [30]); *ii)* $\sigma = 10$, $\mu_I = 0.5$, $\mu_A = 10^{-6}$, $k = 15$, and; *iii)* $\sigma = 1$, $\mu_I = 1$, $\mu_A = 10^{-5}$, $k = 10$. Thus, the LapSVM classifier with the three different settings are re-denoted as: LapSVM$_1$, LapSVM$_2$ and LapSVM$_3$, respectively. For Anchor-K, Anchor-L and EAnchor, a total of $0.1K$ anchors will be identified from data. The number of the closest anchors, s is set as $s = 3$. The iteration number of local anchor embedding is set to be 10 for Anchor-L [31].

As SeRBIA is originally designed for image classification, in this paper, its image pre-processing and feature extraction modules are removed, only its core IF-THEN rule base and decision-maker are kept for numerical data classification. Its user-controlled parameters are set as $\varphi = 1.1$, $\gamma = 0$ [26]. Note that $\gamma = 0$ means SeRBIA will not learn new classes from unlabelled data. For STHP, its externally controlled parameters are set to be: $\gamma_0 = 1.1$ and $H = 6$ [27]. The chunk size is set as 1000 for both SeRBIA and STHP following the same setting as S^{3}OFIS+.

In addition to the single-model comparative algorithms, the most popular semi-supervised ensemble framework, tri-training [28] is also involved in benchmark comparison. In this example, DT, multi-layered perceptron (MLP) [45] and *k*NN are used as base learners, respectively, resulting in three different ensemble models denoted as: TriDT, TriMLP and Tri*k*NN. During the experiments, a three-layered MLP is used and the size of the hidden layer is set as 64 neurons. *k*NN uses $k = 5$. The hyper-parameters of MLP are trained with the gradient descent algorithm. The numerical experiments are performed in offline scenarios for TriDT, TriMLP and Tri*k*NN.

In this numerical example, for each dataset, 10% of data is randomly selected to form the labelled training set and the remaining data is used to build the unlabelled training set. The average class proportion of labelled training sets over 10 Monte Carlo simulations are given by Table 6.

Table 6. Class proportion of labelled training sets for performance evaluation

Dataset	Class Proportion
AU	55.1;44.9
GP	28.0;21.2;29.8;10.1;11.0
IS	14.3;14.3;14.3;14.3;14.3;14.3;14.3
MF	10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0
MG	64.8;35.2
OD	76.9;23.1
OR	9.8;10.1;10.0;10.1;10.1;10.0;10.0;10.1;9.8;10.0
PB	89.6;6.0;0.5;1.6;2.2
PR	10.4;10.4;10.4;9.6;10.4;9.6;9.6;10.4;9.6;9.6
SB	10.7;89.3
SE	14.3;14.3;14.3;14.3;14.3;14.3;14.3
SH	10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0;10.0
TE	9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1;9.1
WI	5.4;94.6

The numerical results obtained by S^{3}OFIS+ and the 12 comparative semi-supervised learning approaches on the 14 benchmark datasets are reported in Table 7 in terms of classification accuracy, where the top three classification accuracy rates per dataset are in bold for visual clarity. The average classification accuracy rates (Acc) and execution time costs (t_{exe}) of S^{3}OFIS+ and the 12 competitors over the 14 benchmark problems are depicted in Figs. 6 and 7 for better demonstration.

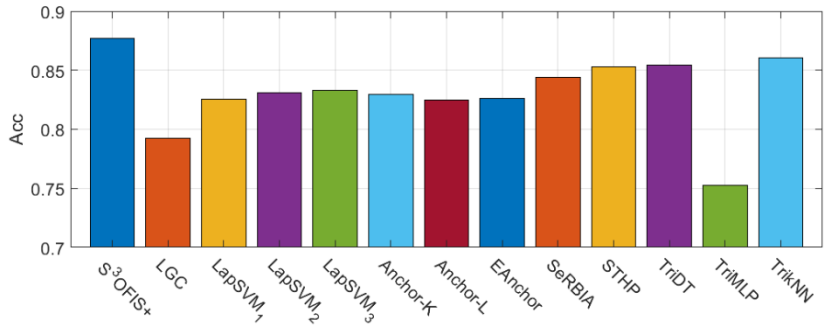


Fig. 6. Average classification accuracies of different semi-supervised classifiers on numerical benchmark datasets

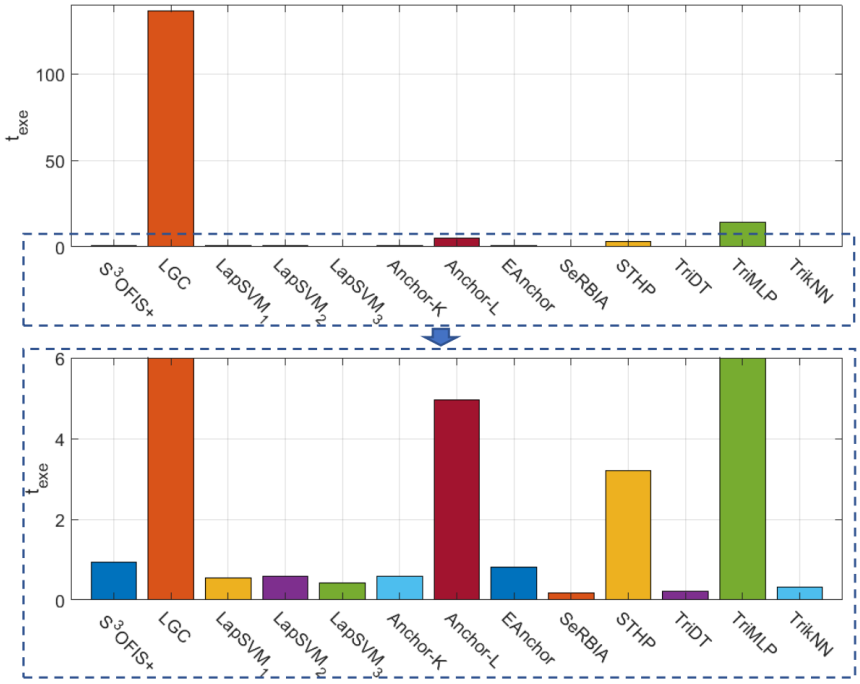


Fig. 7. Average execution time consumptions of different semi-supervised classifiers on numerical benchmark datasets

Table 7. Classification accuracy comparison between different semi-supervised classifiers on numerical benchmark datasets

Algorithm	Dataset						
	AU	GP	IS	MF	MG	OD	OR
S^{3}OFIS+	**0.6019**	**0.7150**	**0.8729**	0.8921	0.7683	**0.9863**	0.9712
LGC	0.5556	**0.7092**	**0.8699**	**0.8981**	0.7415	0.7690	**0.9811**
LapSVM$_1$	0.5557	0.5337	0.8150	0.8977	0.6487	0.9767	0.9757
LapSVM$_2$	0.5729	0.5353	0.8217	**0.8981**	0.6491	0.9729	0.9753
LapSVM$_3$	0.5565	0.6196	0.8355	0.8977	0.6487	0.7695	**0.9767**
Anchor-K	0.5808	0.6539	0.8253	0.8752	0.7048	0.9854	0.9753
Anchor-L	0.5905	0.6286	0.8181	0.8551	0.7051	0.9859	0.9727
EAnchor	0.5950	0.6355	0.8268	0.8873	0.7653	**0.9880**	**0.9810**
SeRBIA	0.5977	0.5804	0.7360	0.8707	0.7043	0.9678	0.9755
STHP	0.5977	0.5995	0.8104	**0.8988**	0.7181	0.9198	0.9765
TriDT	**0.8079**	**0.6563**	**0.9152**	0.8831	**0.8114**	0.9857	0.8104
TriMLP	**0.8288**	0.4739	0.6823	**0.9338**	**0.7861**	0.9655	0.7611
Tri*k*NN	0.5971	0.6470	0.8251	0.8593	**0.7793**	**0.9890**	0.9622
Algorithm	Dataset						
	PB	PR	SB	SE	SH	TE	WI
S^{3}OFIS+	0.9250	**0.9810**	**0.9874**	**0.8631**	0.7910	0.9565	**0.9646**
LGC	0.8981	0.9749	**0.9940**	**0.8548**	0.8186	**0.9761**	0.0540
LapSVM$_1$	0.8981	0.9648	0.9720	0.7968	0.8135	0.7595	0.9460
LapSVM$_2$	0.9059	0.9602	0.9722	0.8008	0.8163	0.8104	0.9460
LapSVM$_3$	0.8981	0.9739	**0.9928**	0.8265	0.8348	0.8826	0.9460
Anchor-K	0.6560	**0.9821**	0.9855	0.8205	0.7799	**0.9633**	0.8243
Anchor-L	0.6905	**0.9810**	0.9762	0.8145	0.7706	0.9558	0.8031
EAnchor	0.6338	**0.9814**	0.9364	0.8185	**0.8382**	0.9617	0.7171
SeRBIA	**0.9452**	0.9721	0.9773	0.7312	**0.8443**	0.9603	0.9506
STHP	0.9088	0.9793	0.9837	0.7939	**0.8366**	**0.9647**	0.9549
TriDT	**0.9524**	0.8823	0.9883	**0.8947**	0.5608	0.8469	**0.9682**
TriMLP	0.9023	0.4698	0.9689	0.6945	0.6484	0.4768	0.9444
Tri*k*NN	**0.9327**	0.9659	0.9784	0.8175	0.7797	0.9499	**0.9596**

To better summarize the key information of Table 7, classification accuracy rates of S^{3}OFIS+ and alternative semi-supervised classifiers on each benchmark dataset are ranked in descending order and the overall ranks are tabulated in Table 8. In addition, the overall ranks of execution time costs of the 13 semi-supervised classifiers over the 14 benchmark datasets from least to most are reported in Table 8 as well.

Table 8. Overall performance ranks of different semi-supervised classifiers from the best to poorest

Algorithm	Overall Rank	
	Acc	t_{exe}
S^{3}OFIS+	**4.25**	7.36
LGC	5.68	11.14
LapSVM$_1$	9.26	3.07
LapSVM$_2$	8.50	2.64
LapSVM$_3$	7.00	2.50
Anchor-K	6.89	4.79
Anchor-L	8.46	10.00
EAnchor	6.21	6.21
SeRBIA	7.39	**2.36**
STHP	6.04	9.43
TriDT	5.29	3.50
TriMLP	9.43	11.36
Tri*k*NN	6.57	3.64

It can be observed from Tables 7 and 8 that S^{3}OFIS+ is able to achieve greater classification accuracy or at least on par with the best performing semi-supervised learning approaches on the majority of the benchmark problems considered in the experiments. Very importantly, the overall classification accuracy rank of S^{3}OFIS+ is the highest, demonstrating its efficacy. Although the computational efficiency of S^{3}OFIS+ is on the average level as one can see from Table 8, its average execution time cost is still below 1 second per dataset, showing that S^{3}OFIS+ is a highly computationally efficient approach for semi-supervised learning.

Next, the classification performance of S^{3}OFIS+ is evaluated on the widely used visual dataset MNIST. During the experiments, 10000, 20000 and 60000 images are randomly selected to build the labelled training set and the rest of the images are used as the unlabelled set. Both labelled and unlabelled training images are converted to 784×1 dimensional vectors for experiments (no feature selection or dimensionality reduction is used). The chunk size L is set as $L = 10000$. The classification accuracy rates of S^{3}OFIS+ on the unlabelled images after being trained under different split ratios are reported in Table 9. For better evaluation, a number of semi-supervised and supervised classification approaches are involved in the experiments for benchmark comparison, which include:

1) SeRBIA classifier [26];
2) SOFIS+ classifier [41];
3) SOFIS classifier [10];
4) Zero-order autonomous learning multi-model (ALMMo0) classifier [11];
5) SVM classifier;
6) DT classifier;
7) *k*NN classifier;
8) Sequence-dictionary-based *k*NN (SD*k*NN) classifier [46];
9) Sequence (SEQ) classifier [46], and;
10) Extreme learning machine (ELM) classifier [47];

LGC, LapSVM, Anchor-K, Anchor-L, EAnchor, STHP, TriDT, TriMLP and Tri*k*NN are not involved in this numerical example. The main reason for excluding them is due to their lower computational efficiency and higher requirement for system memory on large-scale, high-dimensional problems.

During the experiments, the user-controlled parameters of SeRBIA are set as $\varphi = 1.1$, $\gamma = 0$, $W = 10000$; the level of granularity for SOFIS is set to be $G = 12$; SOFIS+ follows the same setting as S^{3}OFIS+, namely, $G = 10$ and $L = 10000$; $k = 5$ is used for *k*NN and SD*k*NN, and; the maximum number of neurons for ELM is set as 200. As SVM, DT, *k*NN, SD*k*NN and ELM are limited to offline applications, all the labelled training data are presented to them at once for training. SeRBIA, SOFIS+ and ALMMo0 are designed for data streaming processing. Thus, training data is presented to SeRBIA and SOFIS+ chunk-by-chunk. Labelled training samples

are presented to ALMMo0 one-by-one due to its different operating mechanism from the other two online algorithms. SOFIS learns from data in a hybrid mode. During the experiment, SOFIS firstly learns from the first 10000 labelled training samples in offline mode. Then, it continues to learn the remaining labelled training samples on a sample-by-sample basis, similar to ALMMo0.

Furthermore, the same experiments are repeated with the 512×1 dimensional feature vectors extracted from the original handwritten digit images using the gist feature descriptor [48]. The numerical results obtained by the 11 algorithms are reported in Table 9 as well. The average classification accuracy rates are also given in the same table. For visual clarity, the best results are in bold. From Table 9 one can see that S^{3}OFIS+ outperforms the alternatives on this visual dataset.

Table 9. Comparison between different classifiers on MNIST dataset

Algorithm	# Labelled Images						Average
	10000		20000		60000		
	Original	Gist	Original	Gist	Original	Gist	
S^{3}OFIS+	**0.9544**	0.9777	**0.9618**	0.9801	**0.9699**	0.9838	**0.9713**
SeRBIA	0.9480	**0.9789**	0.9583	**0.9820**	0.9696	0.9866	0.9706
SOFIS+	0.9483	0.9740	0.9573	0.9786	0.9686	0.9833	0.9684
SOFIS	0.9466	0.9750	0.9557	0.9811	0.9681	0.9865	0.9688
ALMMo0	0.9477	0.9766	0.9575	0.9808	0.9683	0.9864	0.9695
SVM	0.9252	0.9779	0.9322	0.9816	0.9438	0.9857	0.9577
DT	0.8114	0.8454	0.8437	0.8701	0.8779	0.9010	0.8582
*k*NN	0.9470	0.9752	0.9584	0.9797	0.9684	**0.9875**	0.9694
SD*k*NN	0.9473	0.9677	0.9571	0.9745	0.9555	0.9802	0.9637
SEQ	0.9342	0.9646	0.9455	0.9707	0.9646	0.9762	0.9593
ELM	0.1037	0.9046	0.1064	0.9072	0.1453	0.9122	0.5132

Finally, the performance of S^{3}OFIS+ is evaluate on four popular nonstationary synthetic problems (namely, SEA, HYP, PMNIST and RMNIST). The main purpose of this example is to examine its capability to handle concept drifts in data streams. Following the commonly used experimental protocol [37], 1%, 0.8%, 1.4% and 2% of data is randomly selected from the four datasets, respectively, to form the labelled training set and the remaining samples are used to build the unlabelled training set. The ratio and class proportion of labelled training samples of the four datasets are summarized in Table 10. The classification accuracy rates of S^{3}OFIS+ on the four datasets are reported in Table 11, and are further compared with the following state-of-the-art supervised and semi-supervised classifiers: ParsNet [37], stream classification algorithm guided by clustering (SCARGC)-*k*NN [49], SCARGC-SVM [49] and deep evolving denoising autoencoder (DEVDAN) [50]. The best results are in bold for visual clarity. Note that the results by ParsNet, SCARGC-*k*NN, SCARGC-SVM and DEVDAN are obtained directly from [37]. For fair comparison, during the experiments, the chunk size of S^{3}OFIS+ is set as the size of the labelled training set following the experimental protocol used by [37].

Table 10. Ratio and class proportion of labelled training samples of four nonstationary synthetic datasets

Dataset	% Labelled	Class Proportion
SEA	1%	24.4;75.6
Hyperplane	0.8%	52.0;48.0
PMNIST	1.4%	10.3;11.3;9.8;10.3;9.9;8.8;9.9;10.5;9.5;9.6
RMNIST	2%	10.3;11.4;10.4;9.9;9.6;8.8;9.1;10.7;9.5;10.3

Table 11. Comparison on four nonstationary synthetic numerical and visual datasets

Algorithm	SEA	HYP	PMNIST	RMNIST
S^{3}OFIS+	**0.9601**	**0.9081**	**0.8958**	**0.9024**
ParsNet [37]	0.8801	0.8682	0.4625	0.4846
SCARGC-*k*NN [49]	0.7812	0.7820	0.3331	0.2381
SCARGC -SVM [49]	0.8259	0.8189	0.3302	0.2092
DEVDAN	0.7849	0.6256	0.3665	0.2898

One can see from Table 11 that S^{3}OFIS+ outperforms all competitors on the four problems with very limited labelled training samples available, showing its superiority for semi-supervised nonstationary data stream classification.

5.6. Further Evaluations

Further to the above systematic evaluation, the influence of data dimensionality on the performance of S^{3}OFIS+ is investigated. Nine nonstationary synthetic datasets[9] with different dimensionalities varying from 4 to 512 are used for this investigation. The sizes of the synthetic datasets are set to be 2000 uniformly to facilitate computation. During the experiments, 20% of data is randomly selected to build the labelled training set and the remaining data forms the unlabelled training set. In addition, seven previously used semi-supervised classifiers including LGC, LapSVM, Anchor-K, Anchor-L, EAnchor, SeRBIA and STHP are involved for performance comparison under the same experimental protocol. The results by S^{3}OFIS+ and its competitors on the nine datasets are reported in Table 12 in terms of classification accuracy, where the best results are in bold.

Table 12. Comparison on nine nonstationary synthetic datasets with different dimensionalities

Algorithm	#Attributes								Average
	4	8	16	32	64	128	256	512	
S^{3}OFIS+	0.9344	0.8506	0.7491	0.6861	0.5883	0.5744	**0.6013**	**0.6204**	**0.7006**
LGC	0.9299	0.8375	0.5860	0.5551	0.5307	0.5129	0.5090	0.5137	0.6219
LapSVM$_1$	0.9393	0.8899	0.6858	0.6107	0.5525	0.5249	0.5181	0.5053	0.6533
LapSVM$_2$	**0.9575**	**0.9207**	**0.7928**	0.6672	0.5906	0.5447	0.5294	0.5203	0.6904
LapSVM$_3$	0.9475	0.8795	0.6265	0.5669	0.5417	0.5233	0.5167	0.5040	0.6383
Anchor-K	0.9299	0.8410	0.7301	0.6545	0.5939	0.5667	0.5358	0.5215	0.6717
Anchor-L	0.9161	0.8202	0.7071	0.6409	0.5874	0.5591	0.5371	0.5247	0.6616
EAnchor	0.9381	0.8731	0.7740	**0.7017**	**0.6158**	**0.5775**	0.5452	0.5203	0.6932
SeRBIA	0.6918	0.6885	0.6146	0.5801	0.5297	0.5177	0.5093	0.5221	0.5817
STHP	0.6951	0.6911	0.5877	0.5627	0.5380	0.5173	0.5092	0.5221	0.5779

It can be observed from Table 12 that when the dimensionality of the data is greater than 128, S^{3}OFIS+ is able to offer the highest classification accuracy rates well above the runners-up, outperforming all its competitors. This shows that the data dimensionality has a smaller impact on the performance of S^{3}OFIS+ than the alternative approaches involved in this example.

Next, the robustness of S^{3}OFIS+ is examined by adding Gaussian noise to the experimental data. In this example, three previously used benchmark problems, namely, IS, MF and SE are considered. During the experiments, 10dB, 5dB and 0dB additive white Gaussian noise is added onto 20% of data samples selected randomly. Following the same experimental protocol used in Section 5.5, the ratio of labelled samples is set to be 10%. The classification accuracy rates of S^{3}OFIS+, LGC, LapSVM, Anchor-K, Anchor-L, EAnchor, SeRBIA and STHP on the three datasets in noisy environments with different noise intensities are reported Table 13.

Table 13. Comparison on benchmark datasets in noisy environments with different noise intensities

Algorithm	SNR								
	10dB			5dB			0dB		
	IS	MF	SE	IS	MF	SE	IS	MF	SE
S^{3}OFIS+	0.8049	0.8554	0.7871	0.7884	0.8514	0.7806	0.7679	0.8346	0.7603
LGC	0.8094	0.8649	0.7959	0.7927	0.8609	0.7840	0.7736	0.8468	0.7644
LapSVM$_1$	0.7240	0.7320	0.7068	0.7095	0.7137	0.6985	0.6971	0.7090	0.6862
LapSVM$_2$	0.7313	0.7138	0.7121	0.7101	0.7037	0.7003	0.7041	0.7031	0.6857
LapSVM$_3$	0.7164	0.7320	0.7072	0.7124	0.7137	0.6975	0.7032	0.7090	0.6954
Anchor-K	0.7435	0.6821	0.7319	0.7216	0.6533	0.7159	0.6914	0.5827	0.6972
Anchor-L	0.7271	0.6773	0.7165	0.7086	0.6448	0.6980	0.6884	0.6032	0.6826
EAnchor	0.7629	0.7673	0.7518	0.7430	0.7419	0.7424	0.7231	0.6871	0.7140
SeRBIA	0.6523	0.8273	0.6446	0.6503	0.8218	0.6363	0.6318	0.8002	0.6344
STHP	0.7133	0.8693	0.7012	0.6993	0.8590	0.6962	0.6851	0.8484	0.6776

Average classification rates of the eight semi-supervised classifiers on the three benchmark datasets in noisy environments are reported in Table 14. The results are further compared with the baseline results given by Table 6 to allow an insightful understanding of the impact of noise on different semi-supervised classifiers. For the reader's convenience, the average classification accuracy rates by the eight approaches on the three datasets without noise are given in Table 14 as baseline. It is shown by this table that the average classification accuracy

[9] Generated by HyperplaneGenerator API from Scikit-Multiflow

of S^{3}OFIS+ drops by 6.87%, 7.90% and 10.09% with 10dB, 5dB and 0dB Gaussian noise added onto the data, respectively. The percentage decrease of S^{3}OFIS+ is less than the majority of its counterparts (only greater than LGC). Hence, one may conclude from the results that S^{3}OFIS+ has the strong resistance to noise.

Table 14. Investigation of influence of noise intensity on semi-supervised classifiers

Algorithm	Baseline Average	SNR					
		10dB		5dB		0dB	
		Average	Δ %	Average	Δ %	Average	Δ %
S^{3}OFIS+	0.8760	0.8158	-6.87%	0.8068	-7.90%	0.7876	-10.09%
LGC	0.8743	0.8234	-5.82%	0.8125	-7.07%	0.7949	-9.08%
LapSVM$_1$	0.8365	0.7209	-13.82%	0.7072	-15.46%	0.6974	-16.62%
LapSVM$_2$	0.8402	0.7191	-14.41%	0.7047	-16.13%	0.6976	-16.97%
LapSVM$_3$	0.8532	0.7185	-15.79%	0.7079	-17.03%	0.7025	-17.66%
Anchor-K	0.8403	0.7192	-14.41%	0.6969	-17.07%	0.6571	-21.80%
Anchor-L	0.8292	0.7070	-14.74%	0.6838	-17.54%	0.6581	-20.64%
EAnchor	0.8442	0.7607	-9.89%	0.7424	-12.06%	0.7081	-16.13%
SeRBIA	0.7793	0.7081	-9.17%	0.7028	-9.82%	0.6888	-11.61%
STHP	0.8344	0.7613	-8.76%	0.7515	-9.94%	0.7370	-11.67%

Following the numerical example given by Tables 13 and 14, to further test the robustness of S^{3}OFIS+, the same experiments are repeated by adding 0dB additive white Gaussian noise onto 10%, 20% and 40% of unlabelled and labelled samples selected randomly. The obtained results by S^{3}OFIS+ and the seven comparative approaches on the three benchmark datasets with different ratios of noisy samples are tabulated in Table 15. The average classification accuracy rates are given by Table 16, and a comparation with the baseline results is also presented in the same table. The results shown in Tables 15 and 16 further confirm the conclusion made based on Tables 13 and 14. One may also notice from Table 16 that when the ratio of noisy samples is low, i.e., 10%, S^{3}OFIS+ shows even greater resistance to Gaussian noise than LGC. This is thanks to the proposed *C*NP strategy for pseudo-labelling, enabling S^{3}OFIS+ to be more robust towards a small amount of noisy samples.

Table 15. Comparison on benchmark datasets in noisy environments with different ratios of noisy samples

Algorithm	% Noisy Samples								
	10%			20%			40%		
	IS	MF	SE	IS	MF	SE	IS	MF	SE
S^{3}OFIS+	0.8239	0.8638	0.8131	0.7679	0.8346	0.7603	0.6616	0.7862	0.6435
LGC	0.8185	0.8717	0.8056	0.7736	0.8468	0.7644	0.6858	0.8024	0.6708
LapSVM$_1$	0.7429	0.7997	0.7357	0.6971	0.7090	0.6862	0.5995	0.6999	0.5798
LapSVM$_2$	0.7514	0.7961	0.7366	0.7041	0.7031	0.6857	0.5996	0.6961	0.5818
LapSVM$_3$	0.7795	0.7997	0.7609	0.7032	0.7090	0.6954	0.5751	0.6999	0.5623
Anchor-K	0.7511	0.7288	0.7408	0.6914	0.5827	0.6972	0.6009	0.4679	0.5969
Anchor-L	0.7342	0.7018	0.7246	0.6884	0.6032	0.6826	0.5903	0.5161	0.5778
EAnchor	0.7733	0.8043	0.7631	0.7231	0.6871	0.7140	0.6414	0.5650	0.6259
SeRBIA	0.6906	0.8328	0.6771	0.6318	0.8002	0.6344	0.5332	0.7603	0.5258
STHP	0.7446	0.8771	0.7392	0.6851	0.8484	0.6776	0.5694	0.8017	0.5585

Table 16. Investigation of influence of ratio of noisy samples on semi-supervised classifiers

Algorithm	Baseline Average	% Noisy Samples					
		10%		20%		40%	
		Average	Δ %	Average	Δ %	Average	Δ %
S^{3}OFIS+	0.8760	0.8336	-4.84%	0.7876	-10.09%	0.6971	-20.42%
LGC	0.8743	0.8319	-4.85%	0.7949	-9.08%	0.7197	-17.68%
LapSVM$_1$	0.8365	0.7594	-9.22%	0.6974	-16.62%	0.6264	-25.12%
LapSVM$_2$	0.8402	0.7614	-9.38%	0.6976	-16.97%	0.6258	-25.52%
LapSVM$_3$	0.8532	0.7800	-8.58%	0.7025	-17.66%	0.6124	-28.22%
Anchor-K	0.8403	0.7402	-11.91%	0.6571	-21.80%	0.5552	-33.93%
Anchor-L	0.8292	0.7202	-13.15%	0.6581	-20.64%	0.5614	-32.30%
EAnchor	0.8442	0.7802	-7.58%	0.7081	-16.13%	0.6108	-27.65%
SeRBIA	0.7793	0.7335	-5.88%	0.6888	-11.61%	0.6064	-22.19%
STHP	0.8344	0.7870	-5.68%	0.7370	-11.67%	0.6432	-22.91%

Based on the systematic numerical examples presented in this section, it can be concluded that the proposed S^{3}OFIS+ is an effective semi-supervised learning approach for constructing a highly precise classification model from both labelled and unlabelled data. Thanks to its chunk-by-chunk self-training mechanism, S^{3}OFIS+ can be implemented for both online and offline applications, providing users transparent system structure, explainable reasoning and great classification precision with minimum human labelling efforts.

6. Conclusion

This paper introduces a new self-training EIS named S^{3}OFIS+ for data stream classification. By exploiting pseudo labelling via the proposed *C*NP strategy, S^{3}OFIS+ can continuously self-expand its knowledge base and construct finer classification boundaries from unlabelled streaming data on a chunk-by-chunk basis. Numerical examples demonstrate the superiority of S^{3}OFIS+ over alternative classification methods on a wide range of benchmark classification problems, showing the promise of the proposed method as a powerful semi-supervised learning technique for real-world applications.

There are several considerations for future work. First, the optimality of the learned prototypes needs to be investigated. Prototypes play a key role in S^{3}OFIS+ for constructing classification boundaries. However, since S^{3}OFIS+ learns from both labelled and unlabelled data in a non-iterative, single-pass manner, the identified prototypes may lack optimality. It would be interesting to see how S^{3}OFIS+ performs if its prototypes are optimized periodically. Second, the level of granularity needs to be determined by users in the current S^{3}OFIS+. Although it is not a user- or problem-specific parameter and does not need to involve any prior knowledge, this parameter directly determines the level of fineness of the learned classification boundaries of S^{3}OFIS+. It would be very useful to develop an automated approach, enabling the system to self-determine the most suitable granularity setting based on the ensemble properties of streaming data. Nevertheless, this would be highly challenging considering the nonstationary nature of data streams. Last, but not the least, a single-model system is often insufficient to handle large-scale, high-dimensional, high-frequency streaming data. Creating an ensemble framework composed of multiple single-model systems can be very useful for dealing with such situations. However, it would require more efforts to design an ensemble framework specifically for S^{3}OFIS+ to fully exploit its capacity.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] P. Angelov, E. Lughofer, and X. Zhou, "Evolving fuzzy classifiers using different model architectures," *Fuzzy Sets Syst.*, vol. 159, no. 23, pp. 3160–3182, 2008.

[6] 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 Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.

[7] 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.

[8] 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.

[9] 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.

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

[11] P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (ALMMo-0)," in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2017, vol. 2017-May.

[12] D. Ge and X. J. Zeng, "Learning evolving T-S fuzzy systems with both local and global accuracy - a local online optimization approach," *Appl. Soft Comput.*, vol. 86, pp. 795–810, 2018.

[13] 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 aurvey," *Inf. Sci. (Ny).*, vol. 490, pp. 344–368, 2019.

[14] D. Leite, I. Škrjanc, and F. Gomide, "An overview on evolving systems and learning from stream data," *Evol. Syst.*, vol. 11, no. 2, pp. 181–198, 2020.

[15] P. V. de Campos Souza, "Fuzzy neural networks and neuro-fuzzy networks: a review the main techniques and applications used in the literature," *Appl. Soft Comput.*, vol. 92, p. 106275, 2020.

[16] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

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

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

[19] S. R. Safavian and D. Landgrebe, "A survey of decsion tree classifier methodology," *IEEE Trans. Syst. Man. Cybern.*, vol. 21, no. 3, pp. 660–674, 1990.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nat. Methods*, vol. 13, no. 1, pp. 35–35, 2015.

[21] T. Zhu, Y. Lin, and Y. Liu, "Synthetic minority oversampling technique for multiclass imbalance problems," *Pattern Recognit.*, vol. 72, pp. 327–340, 2017.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-paz, "Mixup: beyond empirical risk minimization," in *International conference on learning representations*, 2018, pp. 1–13.

[23] A. Salazar, L. Vergara, and G. Safont, "Generative adversarial networks and Markov random fields for oversampling very small training sets," *Expert Syst. Appl.*, vol. 163, p. 113819, 2021.

[24] K. P. Bennett and A. Demiriz, "Semi-supervised support vector machines," in *Advances in Neural Information Processing Systems*, 1999, pp. 368–374.

[25] Y. F. Li and Z. H. Zhou, "Towards making unlabeled data never hurt," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 175–188, 2015.

[26] X. Gu, P. Angelov, C. Zhang, and P. M. Atkinson, "A semi-supervised deep rule-based approach for complex satellite sensor image analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, p. DOI: 10.1109/TPAMI.2020.3048268, 2020.

[27] X. Gu, "A self-training hierarchical prototype-based approach for semi-supervised classification," *Inf. Sci. (Ny).*, vol. 535, pp. 204–224, 2020.

[28] Z. H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, 2005.

[29] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Adv. Neural. Inform. Process Syst*, 2004, pp. 321–328.

[30] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 2006, pp. 2399–2434, 2006.

[31] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *International Conference on Machine Learning*, 2010, pp. 679–689.

[32] D.-H. Lee, "Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, 2013, p. 2.

[33] I. Triguero, J. A. Sáez, J. Luengo, S. García, and F. Herrera, "On the characterization of noise filters for self-training semi-supervised in nearest neighbor classification," *Neurocomputing*, vol. 132, pp. 30–41, 2014.

[34] U. Maulik and D. Chakraborty, "A self-trained ensemble with semisupervised SVM: an application to pixel classification of remote sensing imagery," *Pattern Recognit.*, vol. 44, no. 3, pp. 615–623, 2011.

[35] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 1, pp. 355–370, 2017.

[36] X. Gu and P. P. Angelov, "Semi-supervised deep rule-based approach for image classification," *Appl. Soft Comput.*, vol. 68, pp. 53–68, 2018.

[37] M. Pratama, A. Ashfahani, and A. Hady, "Weakly supervised deep learning approach in streaming environments," in *IEEE International Conference on Big Data*, 2019, pp. 1195–1202.

[38] M. Das, M. Pratama, J. Zhang, and Y. S. Ong, "A skip-connected evolving recurrent neural network for data stream classification under label latency scenario," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 3717–3724.

[39] M. Das, M. Pratama, and T. Tjahjowidodo, "A self-evolving mutually-operative recurrent network-based model for online tool condition monitoring in delay scenario," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 2775–2783.

[40] M. Pratama, C. Za'in, E. Lughofer, E. Pardede, and D. A. P. Rahayu, "Scalable teacher forcing network for semi-supervised large scale data streams," *Inf. Sci. (Ny).*, vol. 576, pp. 407–431, 2021.

[41] X. Gu, P. Angelov, and Z. Zhao, "Self-organizing fuzzy inference ensemble system for big streaming data classification," *Knowledge-Based Syst.*, vol. 218, p. 106870, 2021.

[42] X. Gu and M. Li, "A multi-granularity locally optimal prototype-based approach for classification," *Inf. Sci. (Ny).*, vol. 569, pp. 157–183, 2021.

[43] V. M. A. Souza and D. F. Silva, "Classification of evolving data streams with infinitely delayed labels," in *IEEE International Conference on Machine Learning and Applications*, 2015, pp. 214–219.

[44] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, 2016.

[45] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Burlin: Springer, 2009.

[46] 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.

[47] 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.

[48] 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.

[49] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *SIAM International Conference on Data Mining 2015, SDM 2015*, 2015, pp. 873–881.

[50] A. Ashfahani, M. Pratama, E. Lughofer, and Y. S. Ong, "DEVDAN: deep evolving denoising autoencoder," *Neurocomputing*, vol. 390, pp. 297–314, 2020.