Semi-Supervised Fuzzily Weighted Adaptive Boosting for Classification

Xiaowei Gu, Plamen P Angelov, *Fellow, IEEE* and Qiang Shen

***Abstract*—Fuzzy systems offer a formal and practically popular methodology for modelling nonlinear problems with inherent uncertainties, entailing strong performance and model interpretability. Particularly, semi-supervised boosting is widely recognised as a powerful approach for creating stronger ensemble classification models in the absence of sufficient labelled data without introducing any modification to the employed base classifiers. However, the potential of fuzzy systems in semi-supervised boosting has not been systematically explored yet. In this study, a novel semi-supervised boosting algorithm devised for zero-order evolving fuzzy systems is proposed. It ensures both the consistence amongst predictions made by individual base classifiers at successive boosting iterations and the respective levels of confidence towards their predictions throughout the process of sample weight updating and ensemble output generation. In so doing, the base classifiers are empowered to gradually focus more on challenging samples that are otherwise hard to generalise, enabling the development of more precise integrated classification boundaries. Numerical evaluations on a range of benchmark problems are carried out, demonstrating the efficacy of the proposed semi-supervised boosting algorithm for constructing ensemble fuzzy classifiers with high accuracy.**

***Index Terms*—boosting; classification; fuzzy system; semi-supervised.**

I. Introduction

SEMI-SUPERVISED learning is a hybridization between supervised and unsupervised learning [1], [2]. Semi-supervised learning goes beyond conventional supervised learning by leveraging a great amount of unlabelled data samples along with a small amount of labelled one to construct a precise classification model, thereby overcoming the labelling bottleneck in light of the scarcity of labelled data. Thanks to the appealing capability of achieving greater classification performance with less human labour, semi-supervised learning has gained growing attentions in recent years and is now one of the most hotly studied topics in the field of machine learning.

Existing semi-supervised learning algorithms [2] can be categorized broadly into two major groups, namely, 1) inductive approaches and 2) transductive approaches. Inductive approaches [3]–[7] generally extend mainstream supervised classification algorithms, such as decision tree (DT), k-nearest neighbours (kNN), support vector machine (SVM), artificial neural network (ANN), etc., to incorporate unlabelled data in model training, yielding a stronger classifier. In contrast, transductive approaches [8]–[11] do not produce a classification model but directly provide predictions on unlabelled data presented during training. Such approaches generally construct a graph structure from both labelled and unlabelled data with samples sharing similar characteristics connected by edges, and then assign labels to unlabelled samples by propagating labelling information along the edges of the graph. Transductive approaches typically offer greater classification performance because their predictions over unlabelled data are directly optimized based on the labelled ones. However, the predictive power of transductive approaches is limited to the unlabelled training samples only because they do not build classification model, whilst the classifiers learned by inductive approaches can be used for classifying any unseen data samples in the same data space [1], [12].

Wrapper methods are among the best known inductive approaches for semi-supervised learning [2], [12] that can practically be used to extend any standard supervised classification algorithms. Wrapper methods, which include self-training [6], [13], co-training [8], [14] and ensemble learning [15]–[18], iteratively train one or multiple supervised base classifiers from labelled training samples and selected unlabelled samples with the most confident predictions made by base classifiers at earlier iterations, namely, pseudo labels.

Different from other wrapper methods, semi-supervised boosting extends conventional supervised boosting algorithms [19]–[21] by utilizing pseudo-labelled samples to augment labelled training data at each boosting iteration. Semi-supervised boosting trains individual base classifiers sequentially with various distributed labelled and pseudo-labelled data through an iterative process. Similar to supervised boosting [22], semi-supervised boosting gives extra sample weights to these difficult-to-classify ones (both labelled and unlabelled) at each iteration to force base classifiers to gradually focus on them, and combines the obtained base classifiers into a stronger ensemble classifier via weighted majority voting, where more weights are given to these more accurate base classifiers. Semi-supervised boosting can effectively improve the classification performance of any supervised classifiers using unlabelled data when there lacks sufficient labelled training data. However, existing studies employ mainstream supervised classifiers as the ensemble components and have reported promising performance, but they pay less attentions to the transparency and explainablity of the constructed ensemble classifiers [15]–[18]. With the wider use of AI models in high-stakes applications in domains such as healthcare, finance, legal, etc.

X. Gu is with the Department of Computer Science, University of Surrey, Guildford, GU2 7XH, UK. email: xiaowei.gu@surrey.ac.uk.

P. Angelov is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK. email: p.angelov@lancaster.ac.uk.

Q. Shen is with the Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK. email: qqs@aber.ac.uk.

Corresponding author: Xiaowei Gu

Manuscript received XXXX XX, 2022; revised XXXX XX, 2022.

[23], [24], transparency and explainablity have become the pressing issues in machine learning research. There is an urgent demand for developing novel semi-supervised ensemble models that offer both great predictive performance and high model interpretability.

Fuzzy systems are widely used for modelling nonlinear problems with real-world uncertainties in the form of easy-to-interpret linguistic IF-THEN fuzzy production rules [25]. They attempt to mimic human reasoning rather than the brain structure, and have regained increasing attention in the current move towards explainable AI [26]–[29]. Evolving fuzzy systems (EFSs) are a class of fuzzy systems that are capable of self-developing and self-updating the system structure and parameters online from data streams in a single-pass manner, efficiently transforming the learned knowledge into human-interpretable fuzzy rules [29]. Zero-order EFSs [30]–[33] are a special group of EFSs that construct a set of prototype-based IF-THEN rules with singleton consequent parts from data to perform fuzzy inference. Compared with fuzzy systems of other types [34]–[37], zero-order EFSs have simpler model structures and lower computational complexity. Their internal reasoning and decision-making are based on the mutual distances of data and, thus, inherently explainable. However, a major issue associated with fuzzy systems (including zero-order EFSs) is that the prediction performance of a fuzzy system is usually limited on high-dimensional, large-scale complex problems without building a oversized rule base. However, system obesity will inevitably reduce the interpretability and efficiency of the learned model [28].

A feasible solution to improve the performance of individual fuzzy systems without trading off their interpretability is to build ensemble models for problems with high complexity. There have been a few ensemble models employing fuzzy systems as ensemble components in the literature, offering greater prediction accuracy. For instance, in [38], [39], an AdaBoost-based approach to construct ensemble systems from neuro-fuzzy systems is proposed. To ensure that all the base models contribute equally in decision-making, an approach to normalize the activity levels of different base models is also proposed in [39]. An ensemble fuzzy rule-based system with attribute regrouping is introduced by [40] to tackle high-dimensional problems. The core idea of this approach is to train multiple base classifiers with different lower dimensional projections of training data to form an ensemble, bypassing the so-called "curse of dimensionality" problem. The very first ensemble fuzzy system using EFSs as base learner is proposed in [41] for data stream classification. An EFS-based ensemble system named parsimonious ensemble (pENsemble) is described in [42]. pENsemble has a flexible evolving ensemble structure that can automatically add and/or prune base models to self-adapt the new patterns of data streams, and is able to perform online attribute selection to reduce computational overheads. An extension of pENsemble, namely, pENsemble+ is proposed by [43]. pENsemble+ can actively select training samples for base classifier updating and is equipped with an ensemble merging mechanism, which merges base classifiers with strong mutual correlation together instead of pruning them to better preserve the diversity between base classifiers. In [44], an online-learning ensemble composed of multiple evolving optimal granular systems (eOGSs) is proposed for time-series prediction. Despite that the base models are all fed with the same data stream, each individual base model uses a different parameter setting from others, and the diversity between base models is thereby attained. The possibility of constructing deep ensemble models with fuzzy systems is firstly explored in [45], resulting in a multi-layered ensemble evolving fuzzy model that can learn multi-layered distributed representations from data for classification. A fuzzily weighted adaptive boosting (FWAdaBoost) algorithm that utilizes confidence scores produced by zero-order EFSs in both weight updating and ensemble output generation to create stronger ensemble evolving fuzzy classifier is introduced in [46]. In [47], an online bagging-based ensemble fuzzy classifier that can autonomous prune base fuzzy classifiers with higher prediction errors is presented. An online sequential ensemble of fuzzy predictors for chunk-wise data stream learning is introduced in [48], which learns a separate fuzzy predictor from each data chunk to maximize the processing speed and reduce system obesity. However, the vast majority of existing studies on ensemble fuzzy systems are conducted in the fully supervised scenarios. The use of fuzzy systems in semi-supervised boosting has not been explored yet.

To bridge this gap, in this paper, a novel semi-supervised boosting algorithm named, Semi-Supervised Fuzzily Weighted Adaptive Boosting (SSFWAdaBoost) is proposed for constructing stronger ensemble fuzzy systems for multi-class classification by using EFSs, in particular, zero-order ones as implementation basis. SSFWAdaBoost extends the recently introduced FWAdaBoost algorithm [46] by incorporating unlabelled samples into the iterative boosting process to construct more precise classification boundaries in the absence of sufficient labelled data. SSFWAdaBoost uses the confidence scores produced by individual base classifiers in sample weight updating for both labelled and pseudo-labelled samples. Its unique sample weight updating scheme forces the base classifiers to gradually focus more on difficult-to-classify labelled samples as well as these unlabelled samples with conflicting predictions produced by individual base classifiers. The utilization of confidence scores in sample weight updating also slows down the weight decline for data samples with low classification margins, especially, these correctly classified ones, enabling the base classifiers to pay sufficient attention to them in later iterations. The consistence between the predicted labels at successive boosting iterations is considered in ensemble output generation such that individual base classifier showing greater consistency with their predecessors will receive higher weights and contribute more in the final ensemble outputs. Numerical studies on benchmark datasets demonstrate that SSFWAdaBoost can construct highly accurate ensemble classifiers with zero-order EFSs as the ensemble components from a combination of labelled and unlabelled data, outperforming the state-of-the-art classification approaches involved in the experimental investigations. It is further demonstrated that SSFWAdaBoost can be utilized to improve the classification performance of first-order EFSs. To summarize, key features of SSFWAdaBoost include:

1) it is a generic semi-supervised boosting algorithm designed specifically for zero-order EFSs to build stronger ensemble models for classification without introducing any modification to the employed system;
2) it integrates both the consistence between the predictions produced by individual base classifiers and their respective levels of confidence in sample weight updating and ensemble output generation;
3) it encourages the base classifiers to focus on more challenging labelled and pseudo-labelled samples whilst paying sufficient attention to these samples with low classification margins, thereby achieving greater classification accuracy.

The remainder of this paper is organized as follows. The theoretical background of this study is provided by Section II. The proposed SSFWAdaBoost is described in Section III and a theoretical analysis on the bound of training error is also given. Numerical examples are presented in Section VI as the proof of concept. This paper is concluded by Section V.

II. Preliminaries

In this section, basic knowledge of zero-order EFSs and FWAdaBoost is recalled briefly as the foundation of this study.

First of all, let $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{L+K}\}$ ($\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, ..., x_{k,M}]^T \in \boldsymbol{R}^M$) be a particular dataset in the M dimensional real data space, $\boldsymbol{R}^M$, where the subscript k denotes the time instance at which $\boldsymbol{x}_k$ is observed. It is assumed that $\mathbf{X}$ is composed of data samples of C different classes. Only the class labels of first L samples, $\mathbf{X}_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L\}$ are available a priori, denoted as $\mathbf{Y}_L = \{y_1, y_2, ..., y_L\}$ ($y_k \in 1, 2, ..., C$). The remaining K data samples, $\mathbf{X}_K = \{\boldsymbol{x}_{L+1}, \boldsymbol{x}_{L+2}, ..., \boldsymbol{x}_{L+K}\}$ are unlabelled. There are $\mathbf{X}_L \cup \mathbf{X}_K = \mathbf{X}$ and $\mathbf{X}_L \cap \mathbf{X}_K = \emptyset$.

A. Zero-order EFSs

Using AnYa type fuzzy rules as example [49], a standard zero-order EFS is composed of C IF-THEN rules in the following form ($c = 1, 2, ..., C$; one rule per class) [29]:

$$\begin{aligned} \boldsymbol{\mathcal{R}}_c:\ & \textit{IF } (\boldsymbol{x} \sim \boldsymbol{a}_{c,1}) \textit{ OR ... OR } (\boldsymbol{x} \sim \boldsymbol{a}_{c,P_c}) \\ & \textit{THEN } (y = c) \end{aligned} \quad (1)$$

where $\boldsymbol{a}_{c,i} = [a_{c,i,1}, a_{c,i,2}, \ldots, a_{c,i,M}]^T$ is the i^{th} premise part (prototype) of $\boldsymbol{\mathcal{R}}_c$; "$\sim$" denotes similarity; P_c is the total number of prototypes associated with $\boldsymbol{\mathcal{R}}_c$.

Prototypes of zero-order EFSs are typically identified from empirically observed data samples through a single-pass, non-iterative process based on their ensemble properties and mutual distances. They help preserve the structure and underlying patterns of data, and form the knowledge bases of the resulting systems for performing reasoning and making inference.

Conventional zero-order EFSs follow the "winner takes all" principle for decision-making based on the confidence scores. Given a particular unlabelled data sample, $\boldsymbol{x}$, each IF-THEN rule (assuming the c^{th} one) of the zero-order EFS will produce a confidence score, denoted as $\lambda_c(\boldsymbol{x})$, namely, one score per rule/class. Although different zero-order EFSs may calculate confidence scores differently [29], $\lambda_c(\boldsymbol{x})$ is usually obtained based on the similarity between $\boldsymbol{x}$ and the prototypes associated with $\boldsymbol{\mathcal{R}}_c$. Taking the self-organising fuzzy inference system (SOFIS) [32], [46] as an example, the confidence score of the c^{th} IF-THEN rule, $\boldsymbol{\mathcal{R}}_c$ on $\boldsymbol{x}$ is derived by Eq. (2):

$$\lambda_c(\boldsymbol{x}) = e^{-||\boldsymbol{x}-\boldsymbol{a}_{c,n^*}||^2} \quad (2)$$

where $||\boldsymbol{x} - \boldsymbol{y}|| = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})}$; $\boldsymbol{a}_{c,n^*}$ denotes the nearest prototype to $\boldsymbol{x}$ in the data space that is associated with $\boldsymbol{\mathcal{R}}_c$; and there is $n^* = \arg\min_{i=1,2,...,P_c}(||\boldsymbol{x} - \boldsymbol{a}_{c,i}||^2)$.

The class label, $\hat{y}$ of $\boldsymbol{x}$ is then determined by the IF-THEN rule with the highest confidence score as follows [29]:

$$\hat{y} = c^*; \quad c^* = \underset{c=1,2,...,C}{\arg\max}\,(\lambda_j(\boldsymbol{x})) \quad (3)$$

B. FWAdaBoost

FWAdaBoost is a recently introduced boosting algorithm designed for constructing stronger fuzzy ensemble classifiers with zero-order FISs as the ensemble components. The algorithmic procedure of FWAdaBoost is summarized by Algorithm 1 [46].

Different from alternative boosting algorithms, FWAdaBoost uses the confidence scores produced by its individual ensemble components in sample weight updating as Eq. (4), such that higher weights are assigned to hard-to-classify samples and these with lower classification margins, resulting in more precise combined classification boundaries [46].

$$w_{t,k} = \frac{w_{t-1,k}e^{-\alpha_t\varphi_{t,k}}}{W_t} \quad (4)$$

where $w_{t,k}$ is the weight of $\boldsymbol{x}_k$ ($\boldsymbol{x}_k \in \mathbf{X}_l$) at the t^{th} iteration; α_t is the weight of the t^{th} base classifier, $h_t(\boldsymbol{x})$; $t = 1, 2, ..., T$; T is the number of iteration; $W_t = \sum_{k=1}^{L} w_{t-1,k}e^{-\alpha_t\varphi_{t,k}}$ is a normalization factor; $\varphi_{t,k}$ is defined by Eq. (5) as the difference between the confidence score corresponding to the true class of $\boldsymbol{x}_k$, namely, $\lambda_{t,y_k}(\boldsymbol{x}_k)$ and the highest confidence score excluding $\lambda_{t,y_k}(\boldsymbol{x}_k)$ produced by $h_t(\boldsymbol{x})$ [46]:

$$\varphi_{t,k} = \lambda_{t,y_k}(\boldsymbol{x}_k) - \max_{\substack{c=1,2,\ldots,C;\\ c\neq y_k}}(\lambda_{t,c}(\boldsymbol{x}_k)) \quad (5)$$

Another key feature of FWAdaBoost is the utilization of confidence scores in its ensemble output generation as given by Eq. (6). In so doing, the higher confident predictions made by the individual ensemble components will play a greater role than these less confident predictions in the final ensemble outputs, leading to greater overall classification accuracy [46].

$$F(\boldsymbol{x}) = \underset{c=1,2,\ldots,C}{\arg\max}\,(\sum_{i=1}^{T}\alpha_i\hat{\varphi}_i\hat{Y}_{i,c}) \quad (6)$$

where $\hat{\varphi}_t$ is defined by Eq. (7) as the difference between the highest and second highest confidence scores produced by $h_t(\boldsymbol{x})$ with respect to $\boldsymbol{x}$, and $\hat{Y}_{t,c}$ is a C-dimensional encoded vector from the predicted label, $\hat{y}_t$ as Eq. (8) [46].

$$\hat{\varphi}_t = \lambda_{t,\hat{y}_t}(\boldsymbol{x}) - \max_{\substack{c=1,2,\ldots,C;\\ c\neq \hat{y}_t}}(\lambda_{t,c}(\boldsymbol{x})) \quad (7)$$

$$\hat{Y}_{t,c} = \begin{cases} 1, & if\ c = \hat{y}_t \\ -\frac{1}{C-1}, & else \end{cases} \quad (8)$$

It is worth mentioning that FWAdaBoost uses only two confidence scores for sample weight updating and ensemble output generation each time. Therefore, it is suitable for both binary and multi-class classification problems.

Algorithm 1 FWAdaBoost.

```
input: training data, $\mathbf{X}_L$ and $\mathbf{Y}_L$; number of iteration, $T$; base classifier, $h(\boldsymbol{x})$.
  for $k = 1$ to $L$ do
    initialize sample weight as: $w_{0,k} = \frac{1}{L}$;
  end for
  for $t = 1$ to $T$ do
    train base classifier $h_t(\boldsymbol{x})$ with weighted $\mathbf{X}_L$ and $\mathbf{Y}_L$;
    use $h_t(\boldsymbol{x})$ to predict the class labels $\hat{\mathbf{Y}}_L$ of $\mathbf{X}_L$;
    calculate the prediction error, $\varepsilon_t$ of $h_t(\boldsymbol{x})$ by Eq. (9);
        $$\varepsilon_t = \sum_{k=1}^{L} w_{t-1,k}\mathbb{I}(\hat{y}_{t,k} \neq y_k) \quad (9)$$
    if $\varepsilon_t < \frac{C-1}{C}$ and $\varepsilon_t > 0$ then
      calculate classifier weight, $\alpha_t$ by Eq. (10);
        $$\alpha_t = \frac{1}{2}(\ln(\frac{1-\varepsilon_t}{\varepsilon_t}) + \ln(C-1)) \quad (10)$$
      for $k = 1$ to $L$ do
        update sample weight, $w_{t,k}$ by Eq. (4);
      end for
    else
      $\alpha_t \leftarrow 0$;
      for $k = 1$ to $L$ do
        $w_{t,k} \leftarrow w_{t-1,k}$;
      end for
    end if
  end for
output: ensemble classifier, $F(\boldsymbol{x})$.
```

III. Proposed SSFWAdaBoost Algorithm

As aforementioned, SSFWAdaBoost is a semi-supervised boosting algorithm designed to construct a strong ensemble fuzzy classifier from both labelled and unlabelled data with minimal requirement of human labelling efforts. In this section, technical details of the proposed SSFWAdaBoost algorithm are presented.

The diagram of SSFWAdaBoost is depicted in Fig. 1. As shown in Fig. 1, SSFWAdaBoost trains a total $T+1$ base classifiers from data, but the first base classifier, $h_0(\boldsymbol{x})$ will not take part in the ensemble output generation. SSFWAdaBoost employs the same ensemble output generating scheme used by FWAdaBoost, such that confidence scores produced by individual ensemble components can be utilized to generate more precise ensemble outputs [46]. However, SSFWAdaBoost further utilizes the unlabelled data samples with the predicted class labels to augment the labelled training set by exploiting the so-called "pseudo labelling" technique [5]. By gradually adjusting the weights of both labelled and unlabelled samples based on the confidence scores produced at each boosting iteration, SSFWAdaBoost is able to focus more on these highly challenging samples and achieve greater classification accuracy with the extra information learned from unlabelled data. Note that, the proposed SSFWAdaBoost is a generic boosting algorithm capable of constructing ensemble fuzzy systems with greater classification accuracy in the absence of sufficient labelled data from different types of zero-order EFSs that calculate the confidence score for each class separately and utilize a similar decision-making scheme as given by Eq. (3) for fuzzy inference. It will be demonstrated in Section VI that SSFWAdaBoost can also be used for boosting the classification performance of multiple-input multiple-output (MIMO) first-order EFSs with only minor modification introduced to the ensemble components for rescaling the value range of system outputs.

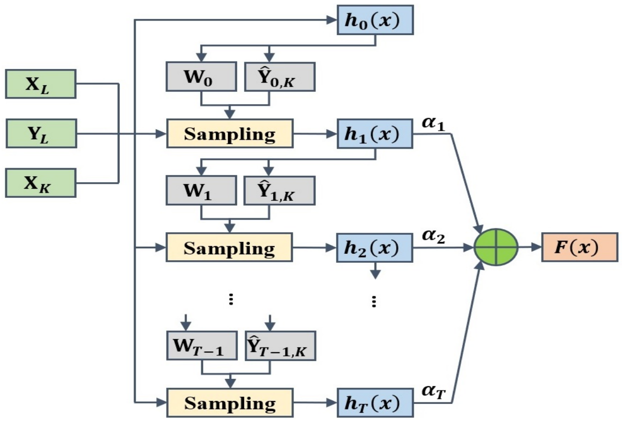


Fig. 1: Diagram of SSFWAdaBoost.

A. Proposed Boosting Scheme

The proposed algorithm is presented in Algorithm 2 and explained in detail as follows.

To incorporate unlabelled data in boosting, a base classifier, $h_0(\boldsymbol{x})$ $(t=0)$ is firstly trained with the labelled data samples, $\mathbf{X}_L$ and $\mathbf{Y}_L$, and used to predict the class labels, $\hat{\mathbf{Y}}_{K,0}$ of unlabelled data samples, $\mathbf{X}_K$ [15]. Then, $\mathbf{X}_L$ and $\mathbf{X}_K$ are combined to build the augmented training set by Eq. (11) with $\hat{\mathbf{Y}}_{K,0}$ used as the pseudo labels of $\mathbf{X}_K$.

$$\mathbf{X}^* \leftarrow \mathbf{X}_L \cup \mathbf{X}_K; \quad \mathbf{Y}_0^* \leftarrow \mathbf{Y}_L \cup \hat{\mathbf{Y}}_{K,0} \tag{11}$$

Note that $h_0(\boldsymbol{x})$ is used for producing the initial pseudo labels for unlabelled samples only. It will not participate in the ensemble output generation.

The weights of data samples within $\mathbf{X}^*$ are then initialized by Eq. (12):

$$w_{0,k} = \begin{cases} \frac{1}{2LW_0}, & if\ \boldsymbol{x}_k \in \mathbf{X}_L \\ \frac{\hat{\varphi}_{0,k}}{2KW_0}, & if\ \boldsymbol{x}_k \in \mathbf{X}_K \end{cases} \tag{12}$$

where $\hat{\varphi}_{0,k}$ is defined by Eq. (7); W_0 is the normalization factor:

$$W_0 = \frac{1}{2} + \frac{1}{2K}\sum_{k=L+1}^{L+K} \hat{\varphi}_{0,k} \tag{13}$$

Remark 1. There are two reasons for initializing the weights of labelled and pseudo-labelled samples differently as Eq. (12).

First, to obtain a meaningful base classifier, labelled samples should play an instrumental role in model training rather than heavily relying on pseudo-labelled samples, especially in the cases where labelled samples are scarce but unlabelled samples are plentiful (namely, $L \ll K$). Second, samples with highly confident pseudo labels should weight more than samples with less confident pseudo labels to reduce the pseudo-labelling errors accumulated in the base classifiers.

Then, the iterative boosting process of SSFWAdaBoost begins ($t \leftarrow t+1$) by training a new base classifier $h_t(\boldsymbol{x})$ with the weighted $\mathbf{X}^*$ with the corresponding class labels, $\mathbf{Y}_t^*$. The base classifier $h_t(\boldsymbol{x})$ is used to predict the class labels, $\hat{\mathbf{Y}}_t$ of $\mathbf{X}^*$ with the prediction error calculated by Eq. (14).

$$\varepsilon_t = \sum_{k=1}^{L+K} w_{t-1,k} \mathbb{I}(\hat{y}_{t,k} \neq y^*_{t-1,k}) \tag{14}$$

where $\hat{y}_{t,k} \in \hat{\mathbf{Y}}_t$ and $y^*_{t-1,k} \in \mathbf{Y}^*_{t-1}$.

Remark 2. By treating the pseudo labels as true labels for these unlabelled samples, the prediction error, ε_t of $h_t(\boldsymbol{x})$ is calculated based on both labelled and pseudo-labelled samples. In so doing, the accordance between pseudo labels generated at the previous boosting iteration and the predicted labels at the current iteration is considered, and these base classifiers that show greater consistency with their predecessors and make less errors on classifying labelled samples will receive higher classifier weights and contribute more in the ensemble outputs.

If the prediction error ε_t satisfies the condition specified by Algorithm 2, the classifier weight, α_t is calculated by Eq. (10), and the sample weights are updated by Eq. (15):

$$w_{t,k} = \begin{cases} \frac{w_{t-1,k} e^{-\alpha_t \varphi_{t,k}}}{W_t}, & if\ \boldsymbol{x}_k \in \mathbf{X}_L \\ \frac{w_{t-1,k} e^{-\alpha_t \hat{\varphi}_{t,k} I_{t,k}}}{W_t}, & if\ \boldsymbol{x}_k \in \mathbf{X}_K \end{cases} \tag{15}$$

where $I_{t,k} = \mathbb{I}(\hat{y}_{t,k} = y^*_{t-1,k}) - \mathbb{I}(\hat{y}_{t,k} \neq y^*_{t-1,k})$; $\varphi_{t,k}$ and $\hat{\varphi}_{t,k}$ are calculated by Eqs. (5) and (7), respectively, and; W_t is defined as:

$$W_t = \sum_{k=1}^{L} w_{t-1,k} e^{-\alpha_t \varphi_{t,k}} + \sum_{k=L+1}^{L+K} w_{t-1,k} e^{-\alpha_t \hat{\varphi}_{t,k} I_{t,k}} \tag{16}$$

Remark 3. The weights of pseudo-labelled samples are updated differently from the weights of labelled samples by Eq. (15) due to the lack of true labels for them. This weight updating scheme gives greater sample weights to hard-to-classify labelled samples and these unlabelled samples with inconsistent pseudo labels over successive boosting iterations, thereby forcing the base classifiers to gradually focus on them.

After both the classifier and sample weights have been updated to the latest, a partial ensemble classifier, $F_t(\boldsymbol{x})$ is built using Eq. (17) to produce a new set of pseudo labels, $\hat{\mathbf{Y}}_{K,t}$ for $\mathbf{X}_K$.

$$F_t(\boldsymbol{x}) = \underset{c=1,2,...,C}{\arg\max} \left(\sum_{i=1}^{t} \alpha_i \hat{\varphi}_i \hat{Y}_{i,c}\right) \tag{17}$$

The class labels of the augmented training set $\mathbf{X}^*$ is then updated as:

$$\mathbf{Y}_t^* \leftarrow \mathbf{Y}_L \cup \hat{\mathbf{Y}}_{K,t} \tag{18}$$

Otherwise, namely, the prediction error, ε_t fails to meet the condition specified by Algorithm 2, the classifier weight of $h_t(\boldsymbol{x})$ is set to be $\alpha_t = 0$, and the class labels of the augmented training set and sample weights stay the same for the next boosting iteration:

$$\mathbf{Y}_t^* \leftarrow \mathbf{Y}^*_{t-1}; \quad w_{t,k} \leftarrow w_{t-1,k}, \forall \boldsymbol{x}_k \in \mathbf{X}^* \tag{19}$$

Once the current iteration finishes, SSFWAdaBoost starts the next boosting iteration until the maximum number of iteration, T is reached.

Algorithm 2 SSFWAdaBoost.

```
input: training data, $\mathbf{X}_L$, $\mathbf{Y}_L$ and $\mathbf{X}_K$; number of iteration, $T$; base classifier, $h(\boldsymbol{x})$.
  train base classifier $h_0(\boldsymbol{x})$ with $\mathbf{X}_L$ and $\mathbf{Y}_L$;
  use $h_0(\boldsymbol{x})$ to predict the class labels $\hat{\mathbf{Y}}_{K,0}$ of $\mathbf{X}_K$;
  obtain augmented training set $\mathbf{X}^*$ and $\mathbf{Y}_0^*$ by Eq. (11);
  for $k = 1$ to $L + K$ do
    initialize sample weight $w_{0,k}$ by Eq. (12);
  end for
  for $t = 1$ to $T$ do
    train base classifier $h_t(\boldsymbol{x})$ with weighted $\mathbf{X}^*$ and $\mathbf{Y}^*_{t-1}$;
    use $h_t(\boldsymbol{x})$ to predict the class labels $\hat{\mathbf{Y}}_t$ of $\mathbf{X}^*_{t-1}$;
    calculate the prediction error, $\varepsilon_t$ of $h_t(\boldsymbol{x})$ by Eq. (14);
    if $\varepsilon_t < \frac{C-1}{C}$ and $\varepsilon_t > 0$ then
      calculate classifier weight, $\alpha_t$ by Eq. (10);
      for $k = 1$ to $L + K$ do
        update sample weight, $w_{t,k}$ by Eq. (15);
      end for
      create partial ensemble classifier $F_t(\boldsymbol{x})$ by Eq. (17);
      use $F_t(\boldsymbol{x})$ to predict the class labels $\hat{\mathbf{Y}}_{K,t}$ of $\mathbf{X}_K$;
      update $\mathbf{Y}^*_{t-1}$ to $\mathbf{Y}^*_t$ by Eq. (18);
    else
      $\alpha_t = 0$;
      $\mathbf{Y}^*_t \leftarrow \mathbf{Y}^*_{t-1}$;
      for $k = 1$ to $L + K$ do
        $w_{t,k} \leftarrow w_{t-1,k}$;
      end for
    end if
  end for
output: ensemble classifier $F(\boldsymbol{x})$.
```

B. Theoretical Justification

The upper bound of the training error (classification error rate on labelled training samples) of SSFWAdaBoost is given by Theorem 1.

Theorem 1: The following upper bound holds for the training error, e_o of the semi-supervised ensemble classifier constructed by SSFWAdaBoost from both labelled and unlabelled training samples:

$$e_o = \frac{1}{L} \sum_{\boldsymbol{x}_k \in \mathbf{X}_L} \mathbb{I}(\hat{y}_k \neq y_k) < W_{T,L} \cdot \prod_{t=0}^{T} W_t \tag{20}$$

where $W_{T,L} = 2(\sum_{\boldsymbol{x}_k \in \mathbf{X}_L} w_{T,k})$.

Detailed proof to Theorem 1 is given by Supplementary Section S1. Note that this proof is derived on the basis of the mathematical proof of Theorem 1 in [46] and is inspired by [20], [21].

Remark 4. Theorem 1 gives the upper bound of the prediction errors made on the labelled training samples by the SSFWAdaBoost-based ensemble systems. However, due to the lack of ground truth for the unlabelled samples, it is not possible to provide an upper bound for the prediction error made on the unlabelled samples. Nevertheless, it will be demonstrated by numerical examples in Section IV that SSFWAdaBoost can effectively improve the classification performance of zero-order EFSs by utilizing a large amount of unlabelled data samples to augment the labelled training set at each boosting iteration via pseudo-labelling.

IV. Experimental Investigation

A. Configuration

In this section, numerical examples on popular benchmark classification problems are presented to demonstrate the efficacy of the proposed SSFWAdaBoost. In this study, three mainstream zero-order EFSs, namely, SOFIS [32], zero-order autonomous learning multi-model system (ALMMo0) [31] and self-organizing fuzzy belief system (SOFIBS) [33] are employed as the base classifiers for demonstrating the peformance of SSFWAdaBoost. For clarity, the ensemble classifiers constructed by SSFWAdaBoost with SOFIS, ALMMo0 and SOFBIS as the ensemble components are denoted as SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS, respectively.

In this study, a total of 18 numerical datasets from UCI Machine Learning Repository[1] and Keel Dataset Repository[2] are employed for performance demonstration, which include 1) Firm teacher (FT); 2) Cardiotocography (CA); 3) German credit (GC); 4) Mammography (MA); 5) Occupancy detection (OD); 6) Wall-following robot navigation (WF); 7) Segment (SE); 8) Letter recognition (LR); 9) Optical recognition of handwritten digits (OR); 10) Pen-based recognition of handwritten digits (PR); 11) Phishing websites (PW); 12) Multiple features (MF); 13) Page-blocks (PB); 14) Shill bidding (SB); 15) Semeion handwritten digit (SH); 16) Texture (TE); 17) Gesture phase segmentation (GP) and 18) Image segmentation (IS). To evaluate the performance of SSFWAdaBoost on high-dimensional problems, four popular visual datasets,namely, MNIST[3], Fashion MNIST (FMNIST)[4], MNIST permutations (PMNIST) and rotations (RMNIST)[5] are employed for experimenting. In running the experiments, each image of the four visual datasets (with the uniform size of 28×28 pixels) is converted into a 784×1 dimensional vector in advance. Key information of the benchmark problems used for performance demonstration is summarized by Supplementary Table S1. Note that BA dataset is employed for visualization purpose only due to its relatively simpler structure and smaller size.

The algorithms are developed using MATLABR2021b platform and experiments are performed on a desktop with dual core i7 processor 3.80 GHz$\times$2 and 32.0 GB RAM. By default, all the reported experimental results are obtained as the average of 10 Monte Carlo experiments in offline scenarios for fair comparison. To allow a certain degree of randomness, in each experiment, the data is divided into different subsets for semi-supervised learning based on the given splitting ratio in a purely random manner. All the numerical examples presented in Sections IV.B, IV.C and IV.D are carried out under the transductive setting and the numerical examples presented in Section IV.E are carried out under the inductive setting. Unless specifically declared otherwise, the number of base classifiers, T is set to be 20 for SSFWAdaBoost following the same setting as [16], [46], [50]. The level of granularity, G for SOFIS is set to be 12 following the recommended setting by [32], [46], the level of granularity, G for SOFBIS is set to be 9 as suggested by [33], and ALMMo0 uses the default setting as given by [31]. Note that these are the recommended parameter settings for SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS, which may perform differently from problem to problem.

B. Sensitivity Analysis

To understand how the number of base classifiers, T influences the performances of the ensemble fuzzy systems built by SSFWAdaBoost, a sensitivity analysis is performed using the following six numerical datasets, namely, FT, CA, GC, MA, OD and WF, and the analysis results are presented in the Supplementary Section S3.

It can be observed from Supplementary Table S2 and Fig. S1 that by increasing the number of base classifiers from 1 to 30, the average classification error rates of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS are reduced from 0.2009 to 0.1802, 0.2300 to 0.1955 and 0.1603 to 0.1573, respectively. In other words, the performances of the three ensemble fuzzy models are improved by 11.49%, 17.65% and 1.91%, respectively. This shows that SSFWAdaBoost is able to construct ensemble fuzzy classifiers with greater accuracy by increasing the number of base classifiers in the ensemble structure. One may also notice that the more base classifiers are created to build the ensemble, the more computational resources the ensemble system consumes, whilst the less performance improvement is observed by further increasing the ensemble scale. In addition, it can be observed from Supplementary Table S3 that by using pseudo-labelled samples to augment the training set at each iteration, the base classifiers are able to identify more prototypes from both labelled and unlabelled training samples and gradually become more focused on these challenging, hard-to-classify samples during the training process. Hence, there is a notable increase in the number of prototypes from the base classifier $h_0(\boldsymbol{x})$ (trained by a smaller amount of labelled samples only) to the base classifier $h_1(\boldsymbol{x})$ (trained by a combination of labelled and pseudo-labelled samples). With the increasing weights given to these hard-to-classify samples over the boosting process, the learned base classifiers (from $h_1(\boldsymbol{x})$ to $h_{30}(\boldsymbol{x})$) will gradually focus on these samples and the numbers of prototypes identified from the augmented training

[1]Available at: https://archive.ics.uci.edu/ml/index.php
[2]Available at: https://sci2s.ugr.es/keel/index.php
[3]Available at: http://yann.lecun.com/exdb/mnist/
[4]Available at: https://github.com/zalandoresearch/fashion-mnist
[5]Available at: https://nlp.stanford.edu/projects/mer/

data will decrease accordingly because of the utilization of weighted sampling with replacement. Due to the same reason, if the base classifier $h_0(\boldsymbol{x})$ is less accurate and makes more pseudo-labelling errors, an increase in the number of prototypes in the base classifiers trained at the first few boosting iterations can be expected because these base classifiers tend to show greater inconsistency in the predictions, causing the weights of some less challenging data samples to increase (for example, see the results obtained using ALMMo0 as the base classifier in Supplementary Table S3). Very importantly, the ensemble systems offer greater classification performances on unlabelled samples surpassing any individual base classifiers (see Supplementary Tables S2 and S3). This further verifies the validity of the proposed semi-supervised boosting algorithm.

It has to be stressed that a globally optimal setting does not exist theoretically. In practice, to maximize the performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS, one may need to adjust the parameter setting according to the nature of data. Nevertheless, it will be demonstrated by the numerical examples presented in the rest of this section that, with the recommended settings, the ensemble fuzzy systems built by SSFWAdaBoost can achieve superior classification performance, surpassing or, at least, on par with alternative single-model and multi-model competitors on 16 benchmark problems different from the ones used for sensitivity analysis.

C. Ablation Analysis

In this section, ablation analysis is performed to demonstrate the effectiveness of the weight initialization and updating schemes designed for SSFWAdaBoost. The classification performances of ensemble fuzzy systems built by SSFWAdaBoost are compared with the ensemble fuzzy systems built by two alternative versions of SSFWAdaBoost, denoted as SSFWAdaBoost$_1$ and SSFWAdaBoost$_2$, respectively. In SSFWAdaBoost$_1$, the initial weights of the pseudo labelled samples are all set to be uniformly the same, $\frac{1}{2KW}$ by removing $\hat{\varphi}_{0,k}$ from Eq. (12). In SSFWAdaBoost$_2$, on top of the modification introduced to SSFWAdaBoost$_1$, the weight updating scheme is further simplified by removing $\varphi_{t,k}$ and $\hat{\varphi}_{t,k}$ from Eq. (15) following the standard approach used by SAMME [21]. The performance of SSFWAdaBoost is compared against SSFWAdaBoost$_1$ and SSFWAdaBoost$_2$ on the same six numerical datasets used for sensitivity analysis under the same experimental settings, and the results are reported in Supplementary Table S4 in terms of average classification error rates on the unlabelled samples per dataset. It can be seen from this table that SSFWAdaBoost is able to offer greater classification accuracy than its two alternatives in all cases with the only exception that SSFWAdaBoost$_1$ outperforms the original design with SOFBIS as the base classifier on FT dataset. This, again, demonstrates the effectiveness of the designed weight initialization and updating schemes.

D. Performance Demonstration under Transductive Setting

In this section, numerical experiments are carried out under the transductive setting for performance evaluation, where the performances of the semi-supervised classifiers are evaluated directly on the unlabelled data used for training [2].

1) Performance comparison against alternative boosting algorithms: First, numerical experiments are conducted to compare the performance of the proposed SSFWAdaBoost against the following eight popular supervised and semi-supervised boosting algorithms: 1) FWAdaBoost [46]; 2) AdaBoost.M1 [19]; 3) AdaBoost.M2 [19]; 4) SAMME [21]; 5) SAMME.R [21]; 6) Robust AdaBoost (RobAdaBoost) [50]; 7) ASSEMBLE [15], and; 8) SemiBoost [16]. Note that ASSEMBLE and SemiBoost are the two most popular semi-boosting algorithms. The 12 numerical datasets, which include SE, LR, OR, PR, PW, MF, PB, SB, SH, TE, GP and IS, are employed for this example. For each dataset, the following four different splitting ratios between labelled and unlabelled samples are considered, namely, 1:19, 1:9, 3:17 and 1:4, which are equivalent to using 5%, 10%, 15% and 20% of data as labelled training samples, respectively. In running the experiments, SOFIS, ALMMo0 and SOFBIS are used as the base classifiers for performance demonstration. The number of base classifiers is set to be $T = 20$ for all the boosting algorithms involved in comparison. SSFWAdaBoost, FWAdaBoost, AdaBoost.M1, AdaBoost.M2, SAMME and SAMME.R do not have any other externally controlled parameters to be predefined. The three parameters of RobAdaBoost, namely, λ, α and β are set as 0.35, 1.5 and 0.5 according to [50]. The parameters of ASSEMBLE are set as $\alpha_i = 1$ and $\beta = 0$ as recommended by [15]. SemiBoost also requires users to preset a scale parameter, σ [16], which has a large impact on the performance of the constructed ensemble classifier. In this study, the value of σ is chosen from the similarity values between data samples at the top 10^{th}, 30^{th} and 50^{th} percentiles following [16], and the the lowest classification error rate on the unlabelled training samples of each dataset is reported. To tackle multi-class problems, RobAdaBoos, ASSEMBLE and SemiBoost use the "one-versus-rest" strategy for decision-making as suggested by [16]. The classification performances of the three EFSs boosted by the nine boosting algorithms over 12 benchmark datasets under different splitting ratios are tabulated in Table I in terms of average classification error rates on the unlabelled training samples, where the best results are in bold. The results obtained by SOFIS, ALMMo0 and SOFBIS are also reported as the baseline. The detailed results (per dataset per splitting ratio) are given by Supplementary Tables S5-S16. For better evaluation, the obtained ensemble classifiers and the single-model EFSs, namely, the baseline, are ranked from the best (1^{st}) to the worst (10^{th}) based on their average classification error rates on each dataset per splitting ratio and the average ranks over the 12 datasets are reported in Supplementary Table S17. In addition, the corresponding classification error rates on the labelled training samples are tabulated in Supplementary Table S18.

From Table I and Supplementary Table S17 one can see that the ensemble classifiers constructed by the proposed SSFWAdaBoost offer the greatest classification performance, outperforming alternative boosting algorithms, including these semi-supervised ones. It is also noticeable that supervised boosting algorithms often fail to offer better results than the

TABLE I
PERFORMANCE COMPARISON BETWEEN SSFWADABOOST AND ALTERNATIVE BOOSTING ALGORITHMS OVER 12 BENCHMARK PROBLEMS UNDER TRANSDUCTIVE SETTING

Algorithm	SOFIS				ALMMo0				SOFBIS			
	1:9	3:17	1:4	1:19	1:9	3:17	1:4	1:19	1:9	3:17	1:4	1:19
SSFWAdaBoost	0.1588	0.1166	**0.0965**	**0.0844**	0.1484	**0.1116**	**0.0934**	**0.0822**	**0.1192**	**0.0908**	**0.0763**	**0.0687**
FWAdaBoost	0.1737	0.1329	0.1105	0.0970	0.1679	0.1296	0.1079	0.0962	0.1389	0.1041	0.0861	0.0764
AdaBoost.M1	0.1890	0.1459	0.1256	0.1097	0.1861	0.1469	0.1284	0.1143	0.1404	0.1030	0.0856	0.0741
AdaBoost.M2	0.2282	0.1717	0.1393	0.1213	0.2174	0.1698	0.1399	0.1234	0.1608	0.1111	0.0903	0.0781
SAMME	0.2088	0.1585	0.1325	0.1146	0.2200	0.1727	0.1517	0.1355	0.1661	0.1229	0.0996	0.0855
SAMME.R	0.2364	0.1568	0.1239	0.1090	0.2416	0.1576	0.1176	0.1014	0.4204	0.4141	0.3815	0.3290
RobAdaBoost	0.1650	0.1251	0.1042	0.0910	0.1650	0.1272	0.1066	0.0939	0.1311	0.0966	0.0796	0.0707
ASSEMBLE	**0.1424**	**0.1112**	0.0974	0.0868	**0.1479**	0.1181	0.1030	0.0925	0.1359	0.1016	0.0848	0.0739
SemiBoost	0.1913	0.1551	0.1368	0.1223	0.1976	0.1660	0.1479	0.1385	0.1789	0.1500	0.1307	0.1193
Baseline	0.1869	0.1468	0.1287	0.1101	0.1793	0.1511	0.1278	0.1190	0.1256	0.0941	0.0799	0.0701

baseline because the base classifiers are insufficiently trained due to the lack of sufficient labelled training samples, leading to the worse performances of the ensemble models. As the semi-supervised boosting algorithms are able to utilize greater amounts of unlabelled samples to augment the labelled training sets, their classification performances are typically greater than the supervised alternatives. In addition, SSFWAdaBoost can utilize the confidence scores produced by the base classifiers for sample weight updating and ensemble output generation (similar to FWAdaBoost), hence, it is able to achieve better classification performance than other boosting algorithms.

To examine the statistical significance of the performance improvement of SSFWAdaBoost over the alternative boosting algorithms and the baseline, Friedman test [53] is firstly carried out using the average classification error rates of different models on each dataset under the four splitting ratios. The test results in terms of p-value are reported in Supplementary Table S19, where one can see that all three p-values returned are 0. For better verification, pairwise Wilcoxon signed rank tests [54] are further carried out as suggested by [55], [56] and the p-value returned from the pairwise tests are reported in Supplementary Table S20. It can be seen from this table that 26 out of the 27 p-values reported are below the level of significance specified by $\alpha = 0.05$, suggesting that the performance of SSFWAdaBoost is significantly better than other comparative boosting algorithms.

2) Performance comparison against alternative semi-supervised classifiers: Next, the classification performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS are compared against a number of well-known semi-supervised approaches on the 12 benchmark problems under the same experimental protocol used in the previous example. In this example, the following eight single-model semi-supervised classifiers are used for comparison: 1) Local and global consistence (LGC) [8]; 2) Greedy gradient max-cut (GGMC) [11]; 3) Anchor graph regularization with kernel weights (AGRK) [10]; 4) Anchor graph regularization with local anchor embedding weights (AGRL) [10]; 5) Efficient anchor graph regularization (EAGR) [51]; 6) Laplacian support vector machine (LSVM) [9]; 7) Transductive minimax probability machine (TMPM) [52], and; 8) Self-training hierarchical prototype-based classifier (STHP) [13]. In running the experiments, the externally controlled parameters, α and μ of LGC and GGMC are set to be 0.99 and 0.01, respectively, as suggested by [8], [11]; and both LGC and GGMC use the kNN graph with $k = 5$; AGRK, AGRL and EAGR select a total of $0.1(L + K)$ anchors from data by k-means ($L + K$ is the total number of data samples); the number of the closest anchors s is set as $s = 3$ [51]; the iteration number of local anchor embedding is set to be 10 for AGRL [10]; as the performance of LSVM is sensitive to its parameter setting, in this study, three different parameter settings are considered for LSVM, namely, *i)* $\sigma = 10$, $\mu_I = 1$, $\mu_A = 10^{-6}$, $k = 15$ (as suggested by [9]); *ii)* $\sigma = 10$, $\mu_I = 0.5$, $\mu_A = 10^{-6}$, $k = 15$; *iii)* $\sigma = 1$, $\mu_I = 1$, $\mu_A = 10^{-6}$, $k = 15$; and the best classification result on the unlabelled training samples of each dataset is reported; the values of two externally controlled parameters, λ and ρ of TMPM are varied from $[10^{-4}, 10^{-3}, 10^{-2}, ..., 10^{4}]$ as suggested by [52], and 10% of the labelled training samples are randomly selected to help TMPM determine the best parameter setting for each problem as the validation samples; STHP uses the following parameter setting $\gamma_o = 1.1$, $H = 6$ and $N = 1000$ [13].

In addition, three mainstream co-training/boosting algorithms are also involved in performance comparison, which include tri-training [3]; ASSEMBLE [15], and; SemiBoost [16]. The following six semi-supervised ensemble models are created using decision tree (DT) and k-nearest neighbour (kNN) classifiers as the base learners, namely, 1) tri-training-based DT ensemble classifier (TriTrain-DT); 2) tri-training-based kNN ensemble classifier (TriTrain-kNN); 3) ASSEMBLE-based DT ensemble classifier (ASSEMBLE-DT); 4) ASSEMBLE-based kNN ensemble classifier (ASSEMBLE-kNN); 5) SemiBoost-based DT ensemble classifier (SemiBoost-DT), and; 6) SemiBoost-based kNN ensemble classifier (SemiBoost-kNN). During the experiments, ASSEMBLE and SemiBoost follow the same protocol used in the previous example; the maximum number of split is set to be $L_t - 1$ for DT (L_t is the number of training samples presented to DT) and $k = 3$ for kNN, following the same setting as [6].

The average classification error rates of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and the 14 single-model and multi-model alternatives on

TABLE II
PERFORMANCE COMPARISON BETWEEN SSFWADABOOST-SOFIS, SSFWADABOOST-ALMMO0, SSFWADABOOST-SOFBIS AND ALTERNATIVE SEMI-SUPERVISED CLASSIFICATION MODELS OVER 12 BENCHMARK PROBLEMS UNDER TRANSDUCTIVE SETTING

Algorithm	1:19	1:9	3:17	1:4
SSFWAdaBoost-SOFIS	0.1588	0.1166	0.0965	0.0844
SSFWAdaBoost-ALMMo0	0.1484	0.1115	0.0934	0.0822
SSFWAdaBoost-SOFBIS	0.1192	**0.0908**	**0.0763**	**0.0687**
LGC	**0.1140**	0.0951	0.0895	0.0843
GGMC	0.2619	0.2289	0.2069	0.1885
AGRK	0.1588	0.1419	0.1314	0.1232
AGRL	0.1635	0.1493	0.1362	0.1295
EAGR	0.1624	0.1417	0.1296	0.1234
LSVM	0.1612	0.1377	0.1137	0.1068
TMPM	0.2053	0.1777	0.1669	0.1573
STHP	0.1588	0.1262	0.1092	0.0997
TriTrain-DT	0.2169	0.1725	0.1482	0.1344
TriTrain-kNN	0.1686	0.1283	0.1047	0.0916
ASSEMBLE-DT	0.1721	0.1025	0.0862	0.0755
ASSEMBLE-kNN	0.1444	0.1079	0.0904	0.0809
SemiBoost-DT	0.1574	0.1502	0.1311	0.1194
SemiBoost-kNN	0.1955	0.1618	0.1433	0.1305

the 12 benchmark problems under four different splitting ratios are reported in Table II (the best results are in bold). Detailed results obtained by theses algorithms are given by Supplementary Tables S21-S24. For better evaluation, the 17 semi-supervised classifiers are ranked from the best (1^{st}) to the worst (17^{th}) based on their classification performances on each dataset per splitting ratio and the average ranks over the 12 datasets are reported in Supplementary Table S25.

It can be observed from Table II and Supplementary Table S25 that the performance of SSFWAdaBoost-SOFBIS surpasses all the alternative semi-supervised classification models in three different experimental settings by offering the lowest classification error rates, apart from being outperformed by LGC under the splitting ratio of 1:19. SSFWAdaBoost-SOFIS and SSFWAdaBoost-ALMMo0 are also among the best-performing models in this performance comparison. In particular, SSFWAdaBoost-ALMMo0 is ranked the respective 5^{th}, 5^{th}, 2^{nd} and 2^{nd} places among the 17 single-model and multi-model semi-supervised classifiers under the four different splitting ratios, whilst SSFWAdaBoost-SOFIS is also ranked the 7^{th}, 6^{th}, 5^{th} and 4^{th} places, respectively. The more labelled training data is presented for training, the better performance the ensemble fuzzy classifiers constructed by SSFWAdaBoost can offer.

The main reason for SSFWAdaBoost-based ensemble fuzzy classifiers to perform worse in the scenario where the labelled data is extremely scarce, i.e., under the splitting ratio 1:19, is that the initial base classifier, $h_o(\boldsymbol{x})$ tends to produce more incorrectly pseudo-labelled samples due to the insufficient amount of labelled training samples. This causes more pseudo-labelling errors to be accumulated in the ensemble classifiers and eventually deteriorate the overall classification performance. For example, it can be seen from Supplementary Table S26 that by randomly correcting $\frac{1}{3}$ of the pseudo-labelling errors made by the initial base classifier $h_o(\boldsymbol{x})$, the overall classification performances of the resulting ensemble classifiers built by SSFWAdaBoost (redenoted as SSFWAdaBoost$^{1/3}$ here) are improved greatly by at least 20%.

To examine the statistical significance, the Friedman test [53] is firstly carried out using the average classification error rates of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and 14 alternative semi-supervised classifiers on each dataset under the four different splitting ratios, where $p = 0$ is returned from the test, showing that the differences between the classification performances of the 17 classifiers are of statistical significance. Next, pairwise Wilcoxon signed rank tests [54] are carried out and the p-values returned from the pairwise tests are reported in Supplementary Table S27, where one can see that 34 out of the 42 p-values reported are below the level of significance specified by $\alpha = 0.05$, suggesting that the performances of SSFWAdaBoost-based ensemble fuzzy classifiers are significantly better.

3) Performance comparison on high-dimensional visual problems: Then, the performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS are evaluated on the four high-dimensional visual datasets, namely, MNIST, FMNIST, PMNIST and RMNIST. To facilitate computation, 10000 images (1000 images per class) are randomly selected from the original dataset and then split into the labelled and unlabelled training sets in running each experiment. In this example, two different splitting ratios are considered, namely, 1:19 and 1:9. The following 15 semi-supervised ensemble models are created using DT, kNN, SOFIS, ALMMo0 and SOFBIS as the base learners for comparison with the same experimental settings used in the previous experiments, which include: 1) TriTrain-DT; 2) TriTrain-kNN; 3) tri-training-based SOFIS ensemble classifier (TriTrain-SOFIS); 4) tri-training-based ALMMo0 ensemble classifier (TriTrain-ALMMo0); 5) tri-training-based SOFBIS ensemble classifier (TriTrain-SOFBIS); 6) ASSEMBLE-DT; 7) ASSEMBLE-kNN; 8) ASSEMBLE-based SOFIS ensemble classifier (ASSEMBLE-SOFIS); 9) ASSEMBLE-based ALMMo0 ensemble classifier (ASSEMBLE-SOFIS); 10) ASSEMBLE-based SOBFIS ensemble classifier (ASSEMBLE-SOFBIS); 11) SemiBoost-DT; 12) SemiBoost-kNN; 13) SemiBoost-based SOFIS ensemble classifier (SemiBoost-SOFIS); 14) SemiBoost-based ALMMo0 ensemble classifier (SemiBoost-ALMMo0), and; 15) SemiBoost-based SOFBIS ensemble classifier (SemiBoost-SOFBIS). The average classification error rates of the 18 semi-supervised ensemble models on the four visual datasets are reported in Table III, where the best results are in bold. The detailed results are reported by Supplementary Table S28 and the average ranks of the 18 ensemble models are presented in Supplementary Table S29. Similar to the previous examples, the Friedman test [53] is firstly carried out to examine whether the differences between the performances of the 18 semi-supervised ensemble classifiers are statistically significant, followed by the pairwise Wilcoxon signed rank tests [54]. The p-value returned from the Friedman test is, again, 0 and the p-values returned from the pairwise tests are reported in Supplementary Table S30.

TABLE III
PERFORMANCE COMPARISON BETWEEN DIFFERENT SEMI-SUPERVISED ENSEMBLE CLASSIFIERS OVER HIGH-DIMENSIONAL PROBLEMS UNDER TRANSDUCTIVE SETTING

Algorithm	1:19	1:9
SSFWAdaBoost-SOFIS	0.1463	0.1141
SSFWAdaBoost-ALMMo0	0.1525	0.1276
SSFWAdaBoost-SOFBIS	0.1693	0.1425
TriTrain-DT	0.3706	0.3148
TriTrain-kNN	0.1906	0.1554
TriTrain-SOFIS	**0.1331**	**0.1102**
TriTrain-ALMMo0	0.1646	0.1389
TriTrain-SOFBIS	0.1788	0.1486
ASSEMBLE-DT	0.2036	0.1748
ASSEMBLE-kNN	0.1606	0.1356
ASSEMBLE-SOFIS	0.1542	0.1177
ASSEMBLE-ALMMo0	0.1782	0.1437
ASSEMBLE-SOFBIS	0.1875	0.1533
SemiBoost-DT	0.2630	0.2293
SemiBoost-kNN	0.2324	0.2035
SemiBoost-SOFIS	0.2029	0.1677
SemiBoost-ALMMo0	0.2091	0.1804
SemiBoost-SOFBIS	0.2255	0.1945

One can see from Table III and Supplementary Table S29 that SSFWAdaBoost-SOFIS and SSFWAdaBoost-ALMMo0 are ranked the second and third places on the four visual datasets among the 18 semi-supervised ensemble classifiers. In addition, it can be observed that SSFWAdaBoost is able to consistently achieve greater performance improvement than the other mainstream co-training/boosting algorithms with SOFIS, ALMMo0 and SOFBIS as its ensemble components, though it is outperformed by tri-training when using SOFIS as the ensemble component. This shows that SSFWAdaBoost can effectively help zero-order EFSs achieve greater prediction performance, which is coincident with the observation made from Table I. The pairwise statistical test results shown in Supplementary Table S30 also suggest that the performance improvement of SSFWAdaBoost is of statistical significance. It is also worth noting that the ensemble classifiers built with ALMMo0 and SOFIS offer better performances than the ones built with SOFBIS because ALMMo0 and SOFIS employ cosine dissimilarity as the distance measure, which is more effective when the dimensionality of data is high.

4) Performance demonstration of SSFWAdaBoost with extremely weak supervision: In this example, numerical experiments are conducted to demonstrate the performance of SSFWAdaBoost in application scenarios with extremely weak supervision, where there only exist a small amounts of labelled training samples and some of the class labels of these labelled samples are also incorrect. In this example, the same 12 datasets used in Table II are employed for demonstration and the splitting ratio between labelled and unlabelled training samples is set as 1:19, namely, only 5% of data samples are labelled. Among these labelled training samples, 5% of them are mislabelled by randomly assigning them to the incorrect classes. The classification performances of the three EFSs, namely, SOFIS, ALMMo0 and SOFBIS boosted by SSFWAdaBoost are reported in Supplementary Table S31, and their original performances are also given in the same table as the baseline results. One can see from this table that SSFWAdaBoost is able to effectively boost the classification performance of the EFSs by creating stronger ensemble classifiers from a small amount of labelled training samples with imperfect ground truth and a great amount of unlabelled samples. The performances of SOFIS, ALMMo0 and SOFBI are improved by 21.46%, 25.94% and 10.86%, respectively with the help of SSFWAdaBoost. This shows the great potential and applicability of SSFWAdaBoost in real-world problems.

5) Demonstration of generality of SSFWAdaBoost: To demonstrate the generality and merits of SSFWAdaBoost, it is applied for boosting the classification performances of three alternative EFSs, which include one zero-order model (eClass0 classifier [30]) and two first-order models (first-order autonomous learning multi-model system, ALMMo1 [57] and self-adaptive fuzzy learning system [58], SAFL). In this example, the same experimental protocols as Table II are used. eClass0, SAFL and ALMMo1 follow the same parameter settings given by [30], [57], [58], respectively. As both SAFL and ALMMo1 are multiple-input single-output EFSs designed for regression problems, the "one-versus-rest" strategy is used in this example for multi-class classification problems [58], and a sigmoid function is utilized to rescale the system outputs to the range of $[0, 1]$ such that the outputs of first-order EFSs can be used as the confidence scores. The classification performances of the three EFSs boosted by SSFWAdaBoost and FWAdaBoost are reported in Supplementary Table S32, and their original performances are also given in the same table as the baseline results. The detailed results are reported in Supplementary Tables S33-S35.

It can be observed from Supplementary Table S32 that SSFWAdaBoost improves the overall classification performances of eClass0, SAFL and ALMMo1 by up to 16.8%, 18.5% and 39.1%, respectively, over the 12 benchmark problems. This example demonstrates the efficacy of SSFWAdaBoost as a generic semi-supervised boosting algorithm for EFSs in the application scenarios where there only exist a small amount of labelled training samples. However, one may also notice that SSFWAdaBoost is less effective than FWAdaBoost on eClass0 and ALMMo1. This is due to the issue that eClass0 and ALMMo1 produce too many pseudo-labelling errors at the initial stage of SSFWAdaBoost, deteriorating the overall boosting performance. Similar to the numerical example presented in Supplementary Table S26, by randomly correcting $\frac{1}{3}$ of the pseudo-labelling errors made by the initial base classifier $h_o(\boldsymbol{x})$, the classification performances of eClass0 and ALMMo1 boosted by SSFWAdaBoost$^{1/3}$ are further improved by 12.44% and 47.36%, respectively, as demonstrated in Supplementary Table S36. This also suggests that one may consider alternative approaches, i.e., the nearest neighbour [15], to perform pseudo labelling prior to the boosting process when $h_o(\boldsymbol{x})$ fails to achieve satisfactory performance due to insufficient labelled training data.

TABLE IV
PERFORMANCE COMPARISON BETWEEN SSFWADABOOST AND ALTERNATIVE BOOSTING ALGORITHMS OVER 12 BENCHMARK PROBLEMS UNDER INDUCTIVE SETTING

Algorithm	SOFIS				ALMMo0				SOFBIS			
	1:11:8	1:5:4	3:9:8	1:2:2	1:11:8	1:5:4	3:9:8	1:2:2	1:11:8	1:5:4	3:9:8	1:2:2
SSFWAdaBoost	0.1670	0.1220	0.1030	0.0902	0.1561	**0.1177**	**0.0993**	**0.0877**	**0.1285**	**0.0956**	**0.0818**	**0.0712**
FWAdaBoost	0.1867	0.1405	0.1169	0.1023	0.1786	0.1354	0.1150	0.1013	0.1491	0.1082	0.0898	0.0773
AdaBoost.M1	0.1999	0.1539	0.1312	0.1156	0.1918	0.1524	0.1325	0.1186	0.1508	0.1081	0.0891	0.0765
AdaBoost.M2	0.2125	0.1624	0.1389	0.1203	0.2165	0.1625	0.1375	0.1239	0.1616	0.1118	0.0925	0.0802
SAMME	0.2111	0.1583	0.1377	0.1195	0.2109	0.1642	0.1421	0.1301	0.1686	0.1196	0.0985	0.0849
SAMME.R	0.2071	0.1534	0.1269	0.1102	0.1931	0.1446	0.1219	0.1061	0.3520	0.3684	0.3054	0.3209
RobAdaBoost	0.1747	0.1329	0.1096	0.0971	0.1742	0.1335	0.1125	0.1000	0.1409	0.1021	0.0852	0.0736
ASSEMBLE	**0.1504**	**0.1135**	**0.0961**	**0.0847**	**0.1543**	0.1217	0.1034	0.0919	0.1427	0.1057	0.0887	0.0774
SemiBoost	0.2019	0.1578	0.1332	0.1163	0.2148	0.1753	0.1539	0.1391	0.1811	0.1408	0.1205	0.1028

E. Performance Demonstration under Inductive Setting

In this section, numerical experiments are carried out under the inductive setting for performance demonstration, where the performances of the semi-supervised classifiers are evaluated on a separate set of unlabelled data after the semi-supervised learning process is completed [2].

1) Performance comparison against alternative boosting algorithms: Firstly, numerical experiments based on the same 12 benchmark datasets used in the previous examples are conducted to compare the performance of the proposed SSFWAdaBoost against the eight supervised and semi-supervised boosting algorithms: 1) FWAdaBoost [46]; 2) AdaBoost.M1 [19]; 3) AdaBoost.M2 [19]; 4) SAMME [21]; 5) SAMME.R [21]; 6) Robust AdaBoost (RobAdaBoost) [50]; 7) ASSEMBLE [15], and; 8) SemiBoost [16]. For each dataset, the following four different splitting ratios between labelled training samples, unlabelled training samples and unlabelled testing samples are considered, namely, 1:11:8, 1:5:4, 3:9:8 and 1:2:2, which are equivalent to using, respectively, 5%, 10%, 15% and 20% of data as labelled training samples, 55%, 50%, 45% and 40% of data as unlabelled training samples and the remaining 40% of data as unlabelled samples for performance evaluation. In running the experiments, for each boosting algorithm, the number of ensemble components, T is varied from 5 to 30 with the interval of 5, and 10% of the labelled training samples are randomly selected and used as the validation samples for the boosting algorithms to identify the optimal value of T, similar to the protocol used by TMPM in Table II. Other settings stay the same as Table I. The classification performances of the three EFSs boosted by the nine boosting algorithms over 12 benchmark datasets under different splitting ratios are tabulated in Table IV in terms of average classification error rates on the unlabelled testing samples, where the best results are in bold. The detailed results (per dataset per splitting ratio) are given by Supplementary Tables S37-S48. The nine ensemble classifiers are further ranked from the best (1^{st}) to the worst (9^{th}) based on their classification performances on each dataset per splitting ratio and the average ranks over the 12 datasets are reported in Supplementary Table S49.

One can see from Table IV and Supplementary Table S49 that the ensemble fuzzy classifiers built by SSFWAdaBoost outperform the ensemble classifiers built by alternative boosting algorithms on classifying the unlabelled testing samples under the inductive setting in terms of average classification performance rank. Although Table IV shows that ASSEMBLE achieves better average classification accuracy than SSFWAdaBoost when SOFIS is used as the base classifier and when ALMMo0 is used for the split ratio 1:11:8, this is due to the noticeably higher classification accuracy of ASSEMBLE than that of SSFWAdaBoost on SE and IS datasets in these cases as reported in Supplementary Tables S37-S41. It is worth noting, however, that with the same base classifier, SSFWAdaBoost outperforms ASSEMBLE on at least half of the 12 datasets for all split ratios and, hence, the average classification performance ranks of SSFWAdaBoost are consistently higher than ASSEMBLE and other competitors as shown in Supplementary Table S49. This example, again, demonstrates the efficacy of SSFWAdaBoost as a highly effective semi-supervised boosting algorithm designed for zero-order EFSs. Similar to the numerical examples presented in the previous section, Friedman tests [53] and pairwise Wilcoxon signed rank tests [54] are carried out to test the statistical significance of the differences between the performances of the ensemble classifiers built by different boosting algorithms. The p-values returned from the statistical tests are reported in Supplementary Tables S50 and S51. It can be seen from the two tables that all the p-values returned from the Friedman tests are 0 and only one p-value returned from the pairwise tests is above the level of significance specified by $\alpha = 0.05$, suggesting that the performance improvement of SSFWAdaBoost is of statistical significance.

2) Performance comparison against alternative semi-supervised ensemble classifiers: Next, the classification performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS are compared with the six semi-supervised ensemble models constructed by tri-training [3], ASSEMBLE [15] and SemiBoost [16] with DT and kNN as the base classifiers, namely, 1) TriTrain-DT; 2) TriTrain-kNN; 3) ASSEMBLE-DT; 4) ASSEMBLE-kNN; 5) SemiBoost-DT, and; 6) SemiBoost-kNN, under the inductive setting following the same experimental protocol used in Table IV. In running the experiments, the maximum number of split of DT is varied from 25, 50 and $L_t - 1$, and the value of k is varied from 3, 5 and 7 for kNN. The number of ensemble components is varied from 5 to 30 with the interval of 5 for ASSEMBLE-DT, ASSEMBLE-kNN, SemiBoost-DT, and

TABLE V
PERFORMANCE COMPARISON BETWEEN DIFFERENT SEMI-SUPERVISED ENSEMBLE CLASSIFIERS OVER 12 BENCHMARK PROBLEMS UNDER INDUCTIVE SETTING

Algorithm	1:11:8	1:5:4	3:9:8	1:2:2
SSFWAdaBoost-SOFIS	0.1670	0.1220	0.1030	0.0901
SSFWAdaBoost-ALMMo0	0.1561	0.1177	0.0993	0.0877
SSFWAdaBoost-SOFBIS	**0.1285**	**0.0956**	**0.0818**	**0.0712**
TriTrain-DT	0.2175	0.1746	0.1516	0.1389
TriTrain-kNN	0.1801	0.1332	0.1137	0.0986
ASSEMBLE-DT	0.1582	0.1124	0.0952	0.0832
ASSEMBLE-kNN	0.1506	0.1118	0.0955	0.0805
SemiBoost-DT	0.1958	0.1452	0.1219	0.1030
SemiBoost-kNN	0.2041	0.1660	0.1447	0.1265

SemiBoost-kNN. The optimal settings of the nine ensemble models are identified by using 10% of labelled training samples as the validation data, and their performances on the unlabelled testing data are reported in Table V. Similar to the previous example, the detailed results (per dataset per splitting ratio) are given by Supplementary Tables S52-S55, and the average ranks of the nine ensemble classifiers over the 12 datasets are reported in Supplementary Table S56. In addition, Friedman test [53] and pairwise Wilcoxon signed rank tests [54] are carried out to examine whether the performance improvement by SSFWAdaBoost is of statistical significance. The p-value returned from the Friedman test is 0 and the p-values returned from pairwise Wilcoxon signed rank tests are reported in Supplementary Table S57.

One can see from Table V and Supplementary Table 56 that SSFWAdaBoost-SOFBIS is able to outperform all other semi-supervised ensemble classifiers in all cases with the lowest classification error rates, and SSFWAdaBoost-ALMMo0 offers the second best results. It can also be observed from Supplementary Table 57 that 14 out of 18 p-values returned from the pairwise tests are below the level of significance specified by $\alpha = 0.05$. This numerical example further demonstrates the effectiveness and validity of the proposed SSFWAdaBoost as a powerful semi-supervised boosting algorithm for building stronger ensemble fuzzy systems for classification.

To summarize, all the experimental studies carried out so far collectively demonstrate the significant potential of SSFWAdaBoost as a powerful semi-supervised boosting algorithm designed for EFSs. SSFWAdaBoost effectively improves the classification performances of different zero-order and first-order EFSs in the application scenarios where labelled training samples are scarce but unlabelled samples are plentiful. In particular, utilizing SOFIS, ALMMo0 and SOFBIS as the base classifiers, the ensemble models constructed by SSFWAdaBoost offer greater performances on 12 popular numerical benchmark problems and four high-dimensional visual ones, outperforming a wide variety of state-of-the-art single-model and multi-model classification approaches.

V. CONCLUSION

In this paper, a novel semi-supervised boosting algorithm named SSFWAdaBoost is proposed for constructing powerful ensemble fuzzy classifiers from a combination of labelled and unlabelled data. With the unique sample weight updating and ensemble output generation schemes, SSFWAdaBoost is able to construct highly precise classification boundaries by training a series of fuzzy classifiers with an increasing focus on these more challenging labelled/unlabelled samples. Numerical examples demonstrate the superior classification performance of the ensemble classifiers constructed by SSFWAdaBoost over the state-of-the-art alternatives, particularly, in the absence of sufficient labelled training data.

There are several considerations for future work. First, SSFWAdaBoost is currently limited to offline application scenarios despite that the employed zero-order EFSs can learn from data streams. It would be very useful to develop an online version of SSFWAdaBoost such that the ensemble models can be continuously updated from new observations. Second, the quality of initial pseudo labels produced prior to the boosting iterations has a great impact on the performance of the constructed ensemble models by SSFWAdaBoost. To improve the performance of the ensemble classifiers, one may consider using some alternative classification models, i.e., semi-supervised ones (like LGC), to produce the initial pseudo labels, instead of training a base classifier with labelled data only to perform this task. Third, by incorporating a certain evolving scheme to automatically update the rule and attribute weights of individual base models, one can expect a significant improvement in the classification accuracy of the resulting ensemble models. Last, but not the least, although preliminary experiments have demonstrated the strong performance of SSFWAdaBoost when the supervision is extremely weak, the robustness of SSFWAdaBoost to noise needs to be investigated in depth for its wider applicability in real-world applications.

REFERENCES

[1] X. Zhu, "Semi-supervised learning literature survey," 2005.

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

[3] Z. 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.

[4] 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.

[5] D. 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.

[6] D. Wu et al., "Self-training semi-supervised classification based on density peaks of data," *Neurocomputing*, vol. 275, pp. 180–191, 2018.

[7] G. Huang et al., "Semi-supervised and unsupervised extreme learning machines," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2405–2417, 2014.

[8] D. Zhou et al., "Learning with local and global consistency," in *Adv. Neural. Inform. Process Syst.*, 2004, pp. 321–328.

[9] 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.

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

[11] J. Wang, T. Jebara, and S. Chang, "Semi-supervised learning using greedy max-cut," *J. Mach. Learn. Res.*, vol. 14, pp. 771–800, 2013.

[12] I. Triguero, S. Garcia, and F. Herrera, "Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study," *Knowl. Inf. Syst.*, vol. 42, no. 2, pp. 245–284, 2015.

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

[14] S. Qiao et al., "Deep co-training for semi-supervised image recognition," in *European Conference on Computer Vision*, 2018, pp. 135–152.

[15] K. Bennett, A. Demiriz, and R. Maclin, "Exploiting unlabeled data in ensemble methods," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 289–296.

[16] P. Mallapragada et al., "SemiBoost: boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, 2008.

[17] K. Chen and S. Wang, "Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 129–143, 2011.

[18] J. Tanha, "MSSBoost: a new multiclass boosting to semi-supervised learning," *Neurocomputing*, vol. 314, pp. 251–266, 2018.

[19] Y. Freund, R. Schapire, and M. Hill, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, 1996, pp. 148–156.

[20] R. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning,* vol. 37, no. 3, pp. 297–336, 1999.

[21] J. Zhu et al., "Multi-class AdaBoost," *Stat. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[22] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010.

[23] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[24] J. Jimenez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nat. Mach. Intell.*, vol. 2, no. 10, pp. 573–584, 2020.

[25] J. Marin-Blazquez and Q. Shen, "From approximative to descriptive fuzzy classifiers," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 4, pp. 484–497, 2002.

[26] I. Skrjanc et al., "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Inf. Sci. (Ny).*, vol. 490, pp. 344–368, 2019.

[27] J. Garibaldi, "The need for fuzzy AI," *IEEE/CAA J. Autom. Sin.*, vol. 6, no. 3, pp. 610–622, 2019.

[28] A. Barredo Arrieta et al., "Explainable artificial antelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, 2020.

[29] X. Gu et al., "Autonomous learning for fuzzy systems: a review," *Artif. Intell. Rev.*, pp. 1–47, 2022.

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

[31] 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, pp. 1–7.

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

[33] X. Gu, P. Angelov, and Q. Shen, "Self-organizing fuzzy belief inference system for classification," *IEEE Trans. Fuzzy Syst.*, vol. 30, no.12, pp. 5473–5483,2022.

[34] N. 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.

[35] M. Antonelli et al., "Multiobjective evolutionary optimization of type-2 fuzzy rule-based systems for financial data classification," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 2, pp. 249–264, 2017.

[36] H. Huang et al., "Recursive least mean dual p-power solution to the generalization of evolving fuzzy system under multiple noises," *Inf. Sci. (Ny).*, vol. 609, pp. 228–247, 2022.

[37] Z. Yang et al., "Statistically evolving fuzzy inference system for non-Gaussian noises," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 4, pp. 2649–2664, 2022.

[38] R. Scherer, "Designing boosting ensemble of relational fuzzy systems," *Int. J. Neural Syst.*, vol. 20, no. 5, pp. 381–388, 2010.

[39] R. Scherer, "An ensemble of logical-type neuro-fuzzy systems," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13115–13120, 2011.

[40] B. Soua, A. Borgi, and M. Tagina, "An ensemble method for fuzzy rule-based classification systems," *Knowl. Inf. Syst.*, vol. 36, no. 2, pp. 385–410, 2013.

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

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

[43] M. Pratama et al., "Online tool condition monitoring based on parsimonious ensemble+," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 664–677, 2020.

[44] D. Leite and I. Skrjanc, "Ensemble of evolving optimal granular experts, OWA aggregation, and time series prediction," *Inf. Sci. (Ny).*, vol. 504, pp. 95–112, 2019.

[45] X. Gu, "Multilayer ensemble evolving fuzzy inference system," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 8, pp. 2425–2431, 2021.

[46] X. Gu and P. Angelov, "Multi-class fuzzily weighted adaptive boosting-based self-organising fuzzy inference ensemble systems for classification," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 9, pp. 3722–3735, 2022.

[47] E. Lughofer, M. Pratama, and I. Skrjanc, "Online bagging of evolving fuzzy systems," *Inf. Sci. (Ny).*, vol. 570, pp. 16–33, 2021.

[48] E. Lughofer and M. Pratama, "Online sequential ensembling of predictive fuzzy systems," *Evol. Syst.*, vol. 13, no. 2, pp. 361–386, 2022.

[49] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *Int. J. Gen. Syst.*, vol. 41, no. 2, pp. 163–185, 2012.

[50] H. Xing and W. Liu, "Robust AdaBoost based ensemble of one-class support vector machines," *Inf. Fusion*, vol. 55, pp. 45–58, 2020.

[51] M. Wang et al., "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, 2016.

[52] G. Huang and C. Du, "The high separation probability assumption for semi-supervised learning," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol.52, no.12, pp. 7561–7573,2022.

[53] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Stat.*, vol. 11, no. 1, pp. 86–92, 1940.

[54] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*, New York: Springer, 1992, pp. 196–202.

[55] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[56] A. Benavoli, G. Corani, and F. Mangili, "Should we really use post-hoc tests based on mean-ranks?," *J. Mach. Learn. Res.*, vol. 17, pp. 1–10, 2016.

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

[58] X. Gu and Q. Shen, "A self-adaptive fuzzy learning system for streaming data prediction," *Inf. Sci. (Ny).*, vol. 579, pp. 623–647, 2021.

Semi-Supervised Fuzzily Weighted Adaptive Boosting for Classification

Supplementary Materials

S1. Proof for Theorem 1

The proof for Theorem 1 is as follows.

For a labelled training sample, $\boldsymbol{x}_k \in \mathbf{X}_L$, the following inequality holds if its predicted label $\hat{y}_k$ by Eq. (6) does not match its true label y_k, namely, $\mathbb{I}(\hat{y}_k \neq y_k) = 1$:

$$H(\boldsymbol{x}_k) = f_{y_k}(\boldsymbol{x}_k) - f_{\hat{y}_k}(\boldsymbol{x}_k) = \sum_{t=1}^{T} \alpha_t \hat{\varphi}_{t,k} \hat{Z}_{t,k} < 0 \tag{S1}$$

where $f_c(\boldsymbol{x}_k) = \sum_{t=1}^{T} \alpha_t \hat{\varphi}_{t,k} \hat{Y}_{t,k,c}$; $\hat{Z}_{t,k} = \hat{Y}_{t,k,y_k} - \hat{Y}_{t,k,\hat{y}_k}$.

According to Eq. (8), there is:

$$\hat{Z}_{t,k} = \begin{cases} \frac{C}{C-1}, & if\ \hat{y}_{t,k} = y_k \ \&\ \hat{y}_{t,k} \neq \hat{y}_k \\ -\frac{C}{C-1}, & if\ \hat{y}_{t,k} \neq y_k \ \&\ \hat{y}_{t,k} = \hat{y}_k \\ 0, & if\ \hat{y}_{t,k} \neq y_k \ \&\ \hat{y}_{t,k} \neq \hat{y}_k \end{cases} \tag{S2}$$

Comparing between Eqs. (5) and (7), there is:

$$\begin{cases} \varphi_{t,k} = \hat{\varphi}_{t,k}, & if\ \hat{y}_{t,k} = y_k \\ \varphi_{t,k} \leq -\hat{\varphi}_{t,k}, & if\ \hat{y}_{t,k} \neq y_k \end{cases} \tag{S3}$$

Combining Eqs. (S2) and (S3), the following inequality holds:

$$\frac{C}{C-1} \varphi_{t,k} \leq \hat{\varphi}_{t,k} \hat{Z}_{t,k} \tag{S4}$$

and inequality (S5) also holds, considering $\alpha_t \geq 0$, $\forall t$:

$$\frac{C}{C-1} \sum_{t=1}^{T} \alpha_t \varphi_{t,k} \leq \sum_{t=1}^{T} \alpha_t \hat{\varphi}_{t,k} \hat{Z}_{t,k} < 0 \tag{S5}$$

Hence, there is:

$$\sum_{t=1}^{T} \alpha_t \varphi_{t,k} < 0 \tag{S6}$$

According to the sample weight updating scheme specified by Eq. (15), there is ($\forall \boldsymbol{x}_k \in \mathbf{X}_L$):

$$w_{T,k} = \frac{w_{0,k} e^{-\sum_{t=1}^{T} \alpha_t \varphi_{t,k}}}{\prod_{t=1}^{T} W_t} = \frac{e^{-\sum_{t=1}^{T} \alpha_t \varphi_{t,k}}}{2L \prod_{t=0}^{T} W_t} \tag{S7}$$

where $w_{0,k} = \frac{1}{2LW_0}$.

Consequently, the following inequality can be derived from inequality (S6) and Eq. (S7):

$$\mathbb{I}(\hat{y}_k \neq y_k) < e^{-\sum_{t=1}^{T} \alpha_t \varphi_{t,k}} = 2Lw_{T,k} \prod_{t=0}^{T} W_t \tag{S8}$$

and the upper boundary of e_o is obtained as:

$$e_o = \frac{1}{L} \sum_{x_k \in \mathbf{X}_L} \mathbb{I}(\hat{y}_k \neq y_k) < 2\left(\sum_{x_k \in \mathbf{X}_L} w_{T,k}\right) \cdot \prod_{t=0}^{T} W_t \tag{S9}$$

S2. Data Description

Table S1. Key information of benchmark datasets used for performance demonstration

Dataset	#(Samples)	#(Attributes)	#(Classes)
FT	10800	16	4
CA	2126	21	3
GC	1000	24	2
MA	11183	6	2
OD	20560	5	2
WF	5456	24	4
SE	2310	19	7
LR	20000	16	26
OR	5620	64	10
PR	10992	16	10
PW	11055	30	2
MF	2000	649	10
PB	5472	10	5
SB	6321	9	2
SH	1593	256	10
TE	5500	40	11
GP	9901	17	5
IS	2520	19	7
MNIST	70000	784	10
FMNIST	70000	784	10
PMNIST	70000	784	10
RMNIST	65000	784	10

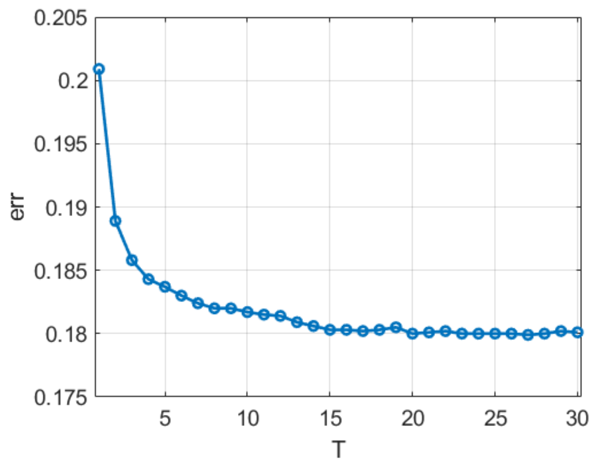
S3. Sensitivity Analysis

In this section, a sensitivity analysis is performed to investigate the influence of the externally controlled parameter, namely, the number of base classifiers, T on the classification performance of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS. The following six datasets, namely, FT, CA, GC, MA, OD and WF are employed for sensitivity analysis. During the experiments, for each dataset, 10% of the data samples are randomly selected to build the labelled training set and the remaining 90% are used as the unlabelled training samples. Note that the experiments are conducted under the tranductive setting and the classification performances of the three ensemble fuzzy systems are evaluated on the unlabelled training sets.

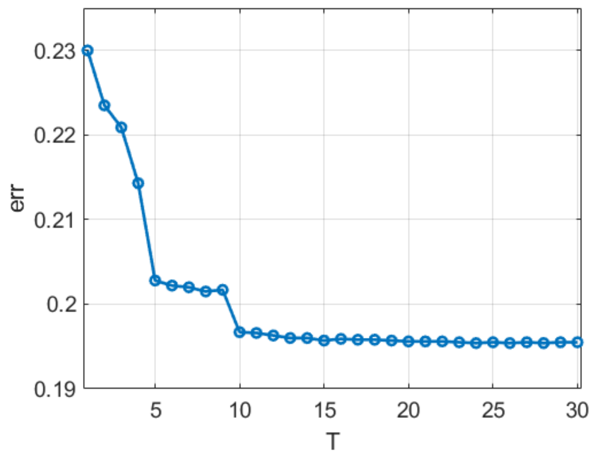
In running the experiments, the value of T is varied from 1 to 30. The three base classifiers, namely, SOFIS, ALMMo0 and SOFBIS use the default parameter settings. The classification error rates (err) of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS on the unlabelled sets of the six datasets with different values of T are reported in Table S2, and the average classification error rates over the six datasets are depicted in Fig. S1. In addition, the average number of prototypes identified by each individual base classifier from the six datasets and the corresponding average classification error rates on the unlabelled training sets are reported in Table S3.

Table S2. Influence of T on classification performances of ensemble fuzzy systems built by SSFWAdaBoost

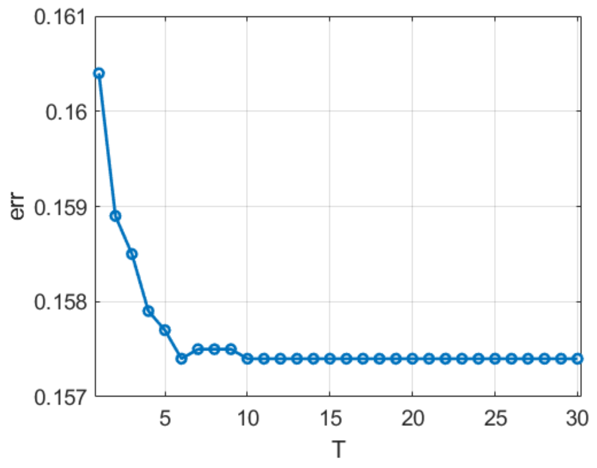
Algorithm	Dataset	T						
		1	5	10	15	20	25	30
SOFIS	FT	0.3383	0.3125	0.3060	0.3029	0.3009	0.3004	0.3002
	CA	0.1694	0.1600	0.1577	0.1557	0.1542	0.1551	0.1558
	GC	0.3661	0.3494	0.3479	0.3457	0.3463	0.3463	0.3456
	MA	0.0584	0.0252	0.0234	0.0230	0.0231	0.0230	0.0230
	OD	0.0200	0.0147	0.0144	0.0143	0.0143	0.0142	0.0143
	WF	0.2534	0.2405	0.2406	0.2402	0.2414	0.2409	0.2420
	Average	0.2009	0.1837	0.1817	0.1803	0.1800	0.1800	0.1802
ALMMo0	FT	0.3909	0.3460	0.3418	0.3409	0.3410	0.3410	0.3411
	CA	0.1746	0.1629	0.1628	0.1617	0.1609	0.1603	0.1605
	GC	0.3583	0.3560	0.3574	0.3551	0.3561	0.3564	0.3568
	MA	0.1821	0.0893	0.0583	0.0577	0.0573	0.0571	0.0570
	OD	0.0253	0.0186	0.0175	0.0171	0.0166	0.0161	0.0159
	WF	0.2487	0.2438	0.2421	0.2416	0.2418	0.2417	0.2416
	Average	0.2300	0.2028	0.1966	0.1957	0.1956	0.1954	0.1955
SOFBIS	FT	0.2996	0.2889	0.2879	0.2879	0.2879	0.2879	0.2878
	CA	0.1445	0.1438	0.1436	0.1432	0.1434	0.1434	0.1434
	GC	0.3184	0.3147	0.3144	0.3143	0.3142	0.3142	0.3143
	MA	0.0201	0.0199	0.0199	0.0199	0.0199	0.0199	0.0199
	OD	0.0137	0.0136	0.0135	0.0135	0.0135	0.0135	0.0135
	WF	0.1658	0.1653	0.1653	0.1653	0.1652	0.1653	0.1651
	Average	0.1603	0.1577	0.1574	0.1574	0.1574	0.1574	0.1573



(a) SOFIS



(b) ALMMo0



(c) SOFBIS

Fig. S1. Average classification errors of ensemble fuzzy systems built by SSFWAdaBoost with different numbers of base classifiers

Table S3. Average numbers of prototypes and classification error rates of individual base classifiers

Base Classifier	SOFIS		ALMMo0		SOFBIS	
	#(Prototypes)	err	#(Prototypes)	err	#(Prototypes)	err
h_0	168.7000	0.2127	246.8500	0.2302	573.4833	0.1601
h_1	438.9333	0.2008	737.9500	0.2241	2011.2170	0.1604
h_2	436.1500	0.2007	757.7000	0.2234	2024.8330	0.1601
h_3	431.1833	0.1956	774.3000	0.2209	1995.2500	0.1600
h_4	422.2667	0.2116	797.5833	0.2110	1868.3670	0.1599
h_5	418.6167	0.2091	823.8333	0.2031	1565.6330	0.1617
h_6	411.4333	0.1953	842.7167	0.2243	1295.8330	0.1661
h_7	407.7667	0.2043	869.6000	0.2142	1064.2830	0.1677
h_8	403.4833	0.2036	899.1667	0.2243	931.7333	0.1688
h_9	402.6167	0.2040	924.3833	0.2099	830.2000	0.1708
h_{10}	399.2833	0.2050	943.1000	0.2212	725.6833	0.1728
h_{11}	399.1000	0.1943	960.8167	0.2208	681.2333	0.1739
h_{12}	395.4333	0.1951	982.6000	0.2243	644.7333	0.1722
h_{13}	387.9667	0.2091	977.3667	0.2254	606.7667	0.1738
h_{14}	382.1833	0.2057	996.9500	0.2095	582.5167	0.1754
h_{15}	376.9333	0.2017	993.6500	0.2254	569.1167	0.1754
h_{16}	370.9167	0.2050	994.7833	0.2206	565.3500	0.1746
h_{17}	361.6667	0.2012	983.5500	0.2142	539.2333	0.1769
h_{18}	354.4667	0.1971	977.4500	0.2155	527.8833	0.1760
h_{19}	347.1333	0.2020	962.6333	0.2149	528.1667	0.1764
h_{20}	338.4333	0.2074	954.9167	0.2211	508.9167	0.1744
h_{21}	332.8667	0.1993	939.0500	0.2272	498.5500	0.1763
h_{22}	325.6167	0.1980	926.7000	0.2183	497.4000	0.1768
h_{23}	318.2167	0.1999	911.3167	0.2222	493.2500	0.1777
h_{24}	314.9000	0.1980	906.9000	0.2216	493.2000	0.1767
h_{25}	308.3500	0.2054	881.6333	0.2141	494.6000	0.1784
h_{26}	305.2333	0.2029	870.7167	0.2236	486.4167	0.1766
h_{27}	299.8667	0.1986	865.7000	0.2202	485.4333	0.1780
h_{28}	295.1667	0.2073	848.2667	0.2151	480.5500	0.1771
h_{29}	293.1500	0.2061	827.9833	0.2157	474.3000	0.1794
h_{30}	289.8000	0.2055	826.1500	0.2217	479.5167	0.1775

S4. Ablation Analysis

Table S4. Ablation analysis results

Base Classifier	**Algorithm**	**FT**	**CA**	**GC**	**MA**	**OD**	**WF**
SOFIS	SSFWAdaBoost	0.3009	0.1542	0.3463	0.0231	0.0143	0.2414
	SSFWAdaBoost$_1$	0.3179	0.1764	0.3479	0.1161	0.0274	0.2601
	SSFWAdaBoost$_2$	0.3303	0.1821	0.3549	0.0585	0.0282	0.2609
ALMMo0	SSFWAdaBoost	0.3410	0.1609	0.3561	0.0573	0.0166	0.2418
	SSFWAdaBoost$_1$	0.3617	0.1881	0.3746	0.1541	0.0291	0.2573
	SSFWAdaBoost$_2$	0.3654	0.1928	0.3714	0.0945	0.0301	0.2575
SOFBIS	SSFWAdaBoost	0.2879	0.1434	0.3142	0.0199	0.0135	0.1652
	SSFWAdaBoost$_1$	0.2859	0.1443	0.3169	0.0200	0.0136	0.1660
	SSFWAdaBoost$_2$	0.2910	0.1441	0.3222	0.0200	0.0137	0.1669

S5. Experimental Results under Transductive Setting

Table S5. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:19

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2800	0.2207	0.0341	0.0349	0.1247	0.1441
FWAdaBoost	0.3279	0.2515	0.0579	0.0548	0.1094	0.1604
AdaBoost.M1	0.3449	0.2941	0.0581	0.0515	0.1281	0.1841
AdaBoost.M2	0.4594	0.3760	0.0534	0.0521	0.1140	0.2095
SAMME	0.3787	0.3421	0.0734	0.0572	0.1276	0.2341
SAMME.R	0.3668	0.2891	0.0582	0.0580	0.1129	0.1958
RobAdaBoost	0.2984	0.2482	0.0484	0.0446	0.1069	0.1577
ASSEMBLE	0.1885	0.2181	0.0359	0.0418	0.1165	0.1472
SemiBoost	0.2178	0.2582	0.0357	0.0361	0.1037	0.2226
Baseline	0.3432	0.3022	0.0521	0.0435	0.1293	0.1858
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0526	0.0245	0.2962	0.0437	0.3744	0.2760
FWAdaBoost	0.0549	0.0294	0.2744	0.0507	0.3890	0.3241
AdaBoost.M1	0.0584	0.0293	0.2957	0.0573	0.4251	0.3417
AdaBoost.M2	0.0570	0.0291	0.4256	0.0551	0.4798	0.4278
SAMME	0.0598	0.0294	0.3429	0.0744	0.4240	0.3616
SAMME.R	0.1023	0.0285	0.7936	0.0530	0.4020	0.3763
RobAdaBoost	0.0509	0.0257	0.2566	0.0489	0.3851	0.3082
ASSEMBLE	0.0654	0.0275	0.2764	0.0463	0.3535	0.1912
SemiBoost	0.1289	0.1047	0.2343	0.2438	0.4949	0.2154
Baseline	0.0548	0.0268	0.2512	0.0563	0.4260	0.3722

Table S6. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:9

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2119	0.1474	0.0277	0.0206	0.1012	0.1011
FWAdaBoost	0.2632	0.1679	0.0385	0.0324	0.0973	0.1154
AdaBoost.M1	0.2796	0.1977	0.0395	0.0338	0.1100	0.1286
AdaBoost.M2	0.3788	0.2396	0.0356	0.0337	0.0985	0.1409
SAMME	0.3027	0.2180	0.0444	0.0368	0.1100	0.1522
SAMME.R	0.3059	0.1949	0.0371	0.0332	0.0970	0.1291
RobAdaBoost	0.2370	0.1641	0.0328	0.0273	0.0902	0.1133
ASSEMBLE	0.1585	0.1404	0.0270	0.0241	0.1025	0.1144
SemiBoost	0.1632	0.2003	0.0243	0.0194	0.0947	0.1410
Baseline	0.3064	0.2069	0.0405	0.0364	0.1129	0.1350
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0447	0.0157	0.1813	0.0268	0.3077	0.2128
FWAdaBoost	0.0483	0.0185	0.2033	0.0312	0.3291	0.2499
AdaBoost.M1	0.0513	0.0210	0.2149	0.0371	0.3693	0.2683
AdaBoost.M2	0.0483	0.0203	0.2397	0.0337	0.4191	0.3720
SAMME	0.0517	0.0212	0.2495	0.0548	0.3686	0.2919
SAMME.R	0.1019	0.0198	0.3017	0.0318	0.3435	0.2862
RobAdaBoost	0.0448	0.0175	0.1904	0.0305	0.3239	0.2291
ASSEMBLE	0.0526	0.0170	0.2050	0.0294	0.2988	0.1648
SemiBoost	0.0965	0.1023	0.1721	0.2100	0.4786	0.1590
Baseline	0.0498	0.0207	0.1989	0.0335	0.3661	0.2544

Table S7. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 3:17

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1754	0.1117	0.0218	0.0159	0.0903	0.0861
FWAdaBoost	0.2091	0.1277	0.0305	0.0236	0.0862	0.0983
AdaBoost.M1	0.2433	0.1515	0.0316	0.0243	0.1010	0.1075
AdaBoost.M2	0.3026	0.1738	0.0275	0.0249	0.0874	0.1105
SAMME	0.2552	0.1656	0.0369	0.0254	0.1010	0.1127
SAMME.R	0.2398	0.1430	0.0294	0.0251	0.0867	0.1113
RobAdaBoost	0.1967	0.1242	0.0256	0.0204	0.0827	0.0941
ASSEMBLE	0.1542	0.1057	0.0215	0.0181	0.0921	0.0971
SemiBoost	0.1346	0.1658	0.0210	0.0150	0.0883	0.1279
Baseline	0.2561	0.1516	0.0325	0.0279	0.0933	0.1241
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0435	0.0138	0.1397	0.0207	0.2705	0.1686
FWAdaBoost	0.0454	0.0168	0.1658	0.0254	0.2917	0.2056
AdaBoost.M1	0.0486	0.0195	0.1881	0.0284	0.3266	0.2372
AdaBoost.M2	0.0460	0.0171	0.1886	0.0271	0.3766	0.2897
SAMME	0.0490	0.0193	0.2117	0.0381	0.3269	0.2485
SAMME.R	0.0748	0.0159	0.1916	0.0258	0.3081	0.2347
RobAdaBoost	0.0438	0.0146	0.1482	0.0237	0.2889	0.1872
ASSEMBLE	0.0514	0.0129	0.1625	0.0222	0.2771	0.1536
SemiBoost	0.0817	0.0985	0.1468	0.1828	0.4520	0.1274
Baseline	0.0477	0.0166	0.1728	0.0304	0.3228	0.2680

Table S8. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1541	0.0924	0.0197	0.0123	0.0791	0.0731
FWAdaBoost	0.1797	0.1025	0.0280	0.0174	0.0791	0.0817
AdaBoost.M1	0.2128	0.1248	0.0282	0.0177	0.0895	0.0916
AdaBoost.M2	0.2684	0.1353	0.0248	0.0178	0.0799	0.0979
SAMME	0.2185	0.1297	0.0353	0.0177	0.0895	0.0914
SAMME.R	0.2142	0.1166	0.0256	0.0184	0.0801	0.0914
RobAdaBoost	0.1634	0.1014	0.0231	0.0151	0.0752	0.0804
ASSEMBLE	0.1472	0.0862	0.0188	0.0133	0.0830	0.0833
SemiBoost	0.1219	0.1454	0.0176	0.0107	0.0824	0.0924
Baseline	0.1943	0.1317	0.0311	0.0179	0.0891	0.0944
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0410	0.0106	0.1240	0.0172	0.2429	0.1460
FWAdaBoost	0.0417	0.0123	0.1539	0.0201	0.2713	0.1758
AdaBoost.M1	0.0475	0.0135	0.1685	0.0241	0.2985	0.1999
AdaBoost.M2	0.0437	0.0128	0.1705	0.0213	0.3450	0.2389
SAMME	0.0475	0.0136	0.1892	0.0348	0.2984	0.2102
SAMME.R	0.0672	0.0114	0.1765	0.0200	0.2882	0.1988
RobAdaBoost	0.0410	0.0109	0.1370	0.0191	0.2637	0.1621
ASSEMBLE	0.0468	0.0096	0.1365	0.0171	0.2601	0.1399
SemiBoost	0.0708	0.0941	0.1387	0.1550	0.4321	0.1069
Baseline	0.0448	0.0119	0.1829	0.0225	0.2918	0.2088

Table S9. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:19

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2603	0.2036	0.0372	0.0315	0.1421	0.1208
FWAdaBoost	0.2899	0.2433	0.0553	0.0459	0.1260	0.1416
AdaBoost.M1	0.3137	0.2869	0.0622	0.0460	0.1348	0.1627
AdaBoost.M2	0.3749	0.3591	0.0521	0.0436	0.1455	0.1782
SAMME	0.3452	0.4450	0.0872	0.0574	0.1356	0.2159
SAMME.R	0.3133	0.2652	0.0511	0.0462	0.1253	0.3014
RobAdaBoost	0.2807	0.2305	0.0462	0.0384	0.1312	0.1465
ASSEMBLE	0.2161	0.2277	0.0345	0.0357	0.1262	0.1403
SemiBoost	0.2218	0.2712	0.0348	0.0397	0.1038	0.2147
Baseline	0.3076	0.2899	0.0470	0.0388	0.1365	0.1484
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0562	0.0229	0.2685	0.0337	0.3539	0.2500
FWAdaBoost	0.0535	0.0283	0.3038	0.0507	0.3940	0.2830
AdaBoost.M1	0.0602	0.0307	0.3211	0.0578	0.4399	0.3175
AdaBoost.M2	0.0598	0.0296	0.4220	0.0564	0.5057	0.3816
SAMME	0.0660	0.0276	0.3890	0.0819	0.4429	0.3462
SAMME.R	0.1023	0.0263	0.8989	0.0505	0.4099	0.3083
RobAdaBoost	0.0531	0.0247	0.2916	0.0476	0.4025	0.2874
ASSEMBLE	0.0606	0.0308	0.2861	0.0471	0.3672	0.2025
SemiBoost	0.1468	0.1046	0.2372	0.2713	0.5030	0.2220
Baseline	0.0514	0.0316	0.2677	0.0490	0.4838	0.2995

Table S10. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:9

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1963	0.1357	0.0286	0.0177	0.1116	0.0903
FWAdaBoost	0.2316	0.1662	0.0362	0.0268	0.1028	0.1070
AdaBoost.M1	0.2633	0.2048	0.0405	0.0278	0.1115	0.1184
AdaBoost.M2	0.3243	0.2364	0.0348	0.0260	0.1141	0.1271
SAMME	0.2803	0.3390	0.0601	0.0395	0.1101	0.1442
SAMME.R	0.2559	0.1796	0.0342	0.0256	0.1024	0.1121
RobAdaBoost	0.2253	0.1546	0.0316	0.0226	0.1056	0.1088
ASSEMBLE	0.1891	0.1454	0.0247	0.0203	0.1073	0.1061
SemiBoost	0.1673	0.2176	0.0242	0.0212	0.0942	0.1565
Baseline	0.2828	0.2115	0.0425	0.0340	0.1106	0.1283
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0466	0.0155	0.1950	0.0222	0.2865	0.1926
FWAdaBoost	0.0452	0.0187	0.2303	0.0307	0.3363	0.2238
AdaBoost.M1	0.0514	0.0222	0.2488	0.0378	0.3840	0.2520
AdaBoost.M2	0.0500	0.0214	0.2740	0.0346	0.4737	0.3208
SAMME	0.0532	0.0212	0.2990	0.0557	0.3907	0.2795
SAMME.R	0.0906	0.0181	0.4478	0.0301	0.3556	0.2395
RobAdaBoost	0.0437	0.0180	0.2186	0.0296	0.3445	0.2230
ASSEMBLE	0.0577	0.0191	0.2144	0.0309	0.3180	0.1838
SemiBoost	0.1207	0.1023	0.1823	0.2490	0.4924	0.1646
Baseline	0.0577	0.0234	0.2080	0.0352	0.4202	0.2588

Table S11. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 3:17

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1569	0.1036	0.0232	0.0147	0.0971	0.0803
FWAdaBoost	0.1874	0.1263	0.0284	0.0204	0.0891	0.0903
AdaBoost.M1	0.2272	0.1611	0.0337	0.0216	0.0956	0.1030
AdaBoost.M2	0.2629	0.1716	0.0256	0.0193	0.0986	0.1018
SAMME	0.2512	0.2851	0.0421	0.0262	0.0965	0.1405
SAMME.R	0.2058	0.1341	0.0252	0.0193	0.0891	0.0964
RobAdaBoost	0.1846	0.1178	0.0249	0.0175	0.0893	0.0924
ASSEMBLE	0.1656	0.1127	0.0201	0.0169	0.0934	0.0925
SemiBoost	0.1458	0.1866	0.0206	0.0158	0.0885	0.1316
Baseline	0.2316	0.1633	0.0283	0.0204	0.0992	0.1106
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0454	0.0130	0.1629	0.0172	0.2540	0.1523
FWAdaBoost	0.0440	0.0169	0.1799	0.0244	0.3069	0.1811
AdaBoost.M1	0.0545	0.0196	0.2120	0.0311	0.3617	0.2199
AdaBoost.M2	0.0492	0.0177	0.2207	0.0285	0.4275	0.2553
SAMME	0.0531	0.0203	0.2544	0.0426	0.3638	0.2450
SAMME.R	0.0780	0.0155	0.1952	0.0240	0.3229	0.2055
RobAdaBoost	0.0460	0.0148	0.1777	0.0228	0.3142	0.1772
ASSEMBLE	0.0551	0.0158	0.1809	0.0248	0.2980	0.1603
SemiBoost	0.0931	0.0991	0.1521	0.2276	0.4739	0.1394
Baseline	0.0617	0.0192	0.1684	0.0291	0.3844	0.2180

Table S12. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1377	0.0872	0.0211	0.0116	0.0826	0.0683
FWAdaBoost	0.1634	0.1044	0.0259	0.0155	0.0797	0.0763
AdaBoost.M1	0.1982	0.1371	0.0302	0.0174	0.0857	0.0922
AdaBoost.M2	0.2328	0.1392	0.0227	0.0148	0.0849	0.0893
SAMME	0.2164	0.2528	0.0386	0.0245	0.0848	0.1139
SAMME.R	0.1774	0.1101	0.0230	0.0144	0.0803	0.0800
RobAdaBoost	0.1524	0.0975	0.0217	0.0131	0.0783	0.0781
ASSEMBLE	0.1508	0.0928	0.0176	0.0131	0.0850	0.0792
SemiBoost	0.1357	0.1701	0.0182	0.0113	0.0820	0.1194
Baseline	0.2105	0.1460	0.0276	0.0174	0.0925	0.0919
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0441	0.0103	0.1474	0.0142	0.2367	0.1253
FWAdaBoost	0.0419	0.0124	0.1678	0.0203	0.2907	0.1557
AdaBoost.M1	0.0524	0.0156	0.1841	0.0266	0.3374	0.1952
AdaBoost.M2	0.0470	0.0140	0.1918	0.0218	0.4030	0.2199
SAMME	0.0524	0.0150	0.2285	0.0409	0.3442	0.2140
SAMME.R	0.0466	0.0122	0.1790	0.0192	0.3057	0.1690
RobAdaBoost	0.0428	0.0113	0.1639	0.0185	0.2976	0.1511
ASSEMBLE	0.0509	0.0124	0.1596	0.0197	0.2850	0.1441
SemiBoost	0.0859	0.0960	0.1481	0.2028	0.4704	0.1218
Baseline	0.0553	0.0156	0.1664	0.0214	0.3628	0.2202

Table S13. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:19

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1263	0.1988	0.0366	0.0332	0.1005	0.0671
FWAdaBoost	0.1439	0.2557	0.0585	0.0428	0.1076	0.0973
AdaBoost.M1	0.1389	0.2222	0.0684	0.0444	0.1055	0.1076
AdaBoost.M2	0.1647	0.2925	0.0590	0.0392	0.1115	0.1124
SAMME	0.1673	0.2474	0.0918	0.0719	0.1054	0.1581
SAMME.R	0.6162	0.3850	0.6584	0.3081	0.2530	0.4334
RobAdaBoost	0.1305	0.2235	0.0563	0.0373	0.1046	0.0897
ASSEMBLE	0.1627	0.2059	0.0398	0.0345	0.1045	0.1090
SemiBoost	0.1746	0.2627	0.0304	0.0285	0.1016	0.1063
Baseline	0.1346	0.2187	0.0510	0.0310	0.1038	0.0974
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0789	0.0253	0.2311	0.0625	0.3403	0.1294
FWAdaBoost	0.0810	0.0260	0.2763	0.0927	0.3460	0.1385
AdaBoost.M1	0.0760	0.0332	0.3115	0.0973	0.3362	0.1434
AdaBoost.M2	0.0796	0.0280	0.3655	0.0929	0.4162	0.1681
SAMME	0.0822	0.0315	0.3793	0.1433	0.3431	0.1724
SAMME.R	0.0991	0.0377	0.8377	0.3363	0.5823	0.4976
RobAdaBoost	0.0771	0.0255	0.2755	0.0797	0.3379	0.1356
ASSEMBLE	0.0830	0.0238	0.2789	0.0613	0.3621	0.1657
SemiBoost	0.1790	0.1040	0.2456	0.2370	0.4952	0.1816
Baseline	0.0750	0.0228	0.2412	0.0601	0.3333	0.1378

Table S14. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:9

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.0981	0.1359	0.0299	0.0213	0.0818	0.0557
FWAdaBoost	0.1126	0.1810	0.0421	0.0274	0.0851	0.0719
AdaBoost.M1	0.0999	0.1516	0.0462	0.0264	0.0845	0.0750
AdaBoost.M2	0.1114	0.1828	0.0405	0.0242	0.0890	0.0688
SAMME	0.1187	0.1885	0.0589	0.0377	0.0845	0.1286
SAMME.R	0.6408	0.3370	0.3212	0.0511	0.1622	0.7374
RobAdaBoost	0.0984	0.1519	0.0405	0.0227	0.0824	0.0636
ASSEMBLE	0.1181	0.1297	0.0306	0.0226	0.0899	0.0729
SemiBoost	0.1297	0.2035	0.0261	0.0194	0.0936	0.0871
Baseline	0.0943	0.1501	0.0417	0.0249	0.0836	0.0517
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0640	0.0142	0.1715	0.0439	0.2758	0.0978
FWAdaBoost	0.0645	0.0138	0.1992	0.0601	0.2796	0.1117
AdaBoost.M1	0.0654	0.0213	0.2264	0.0646	0.2720	0.1023
AdaBoost.M2	0.0699	0.0148	0.2419	0.0569	0.3261	0.1063
SAMME	0.0681	0.0228	0.2708	0.0958	0.2720	0.1291
SAMME.R	0.0982	0.2081	0.8359	0.4754	0.4496	0.6528
RobAdaBoost	0.0650	0.0136	0.1980	0.0522	0.2745	0.0962
ASSEMBLE	0.0711	0.0113	0.2091	0.0426	0.2990	0.1218
SemiBoost	0.1442	0.1011	0.1893	0.1987	0.4799	0.1280
Baseline	0.0703	0.0172	0.1773	0.0545	0.2652	0.0988

Table S15. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 3:17

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.0817	0.1047	0.0246	0.0192	0.0707	0.0444
FWAdaBoost	0.0855	0.1390	0.0345	0.0206	0.0729	0.0616
AdaBoost.M1	0.0835	0.1153	0.0357	0.0190	0.0735	0.0651
AdaBoost.M2	0.0881	0.1349	0.0311	0.0191	0.0758	0.0555
SAMME	0.0991	0.1351	0.0523	0.0286	0.0735	0.0820
SAMME.R	0.6116	0.2869	0.5600	0.2101	0.0872	0.4965
RobAdaBoost	0.0786	0.1170	0.0317	0.0186	0.0719	0.0528
ASSEMBLE	0.0943	0.0985	0.0246	0.0208	0.0783	0.0621
SemiBoost	0.0996	0.1700	0.0215	0.0161	0.0865	0.0675
Baseline	0.0847	0.1142	0.0276	0.0188	0.0761	0.0506
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0643	0.0103	0.1425	0.0346	0.2438	0.0744
FWAdaBoost	0.0596	0.0114	0.1696	0.0473	0.2489	0.0822
AdaBoost.M1	0.0627	0.0169	0.1930	0.0495	0.2375	0.0760
AdaBoost.M2	0.0635	0.0122	0.1982	0.0427	0.2832	0.0798
SAMME	0.0647	0.0147	0.2405	0.0712	0.2375	0.0956
SAMME.R	0.0988	0.1133	0.7811	0.1945	0.4414	0.6962
RobAdaBoost	0.0598	0.0112	0.1638	0.0395	0.2389	0.0719
ASSEMBLE	0.066	0.0098	0.1730	0.0338	0.2653	0.0913
SemiBoost	0.1204	0.0986	0.1618	0.1758	0.4542	0.0970
Baseline	0.0656	0.0141	0.1588	0.0394	0.2329	0.0766

Table S16. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.0702	0.0852	0.0240	0.0146	0.0636	0.0446
FWAdaBoost	0.0744	0.1221	0.0319	0.0170	0.0635	0.0531
AdaBoost.M1	0.0728	0.0946	0.0294	0.0137	0.0636	0.0542
AdaBoost.M2	0.0743	0.1080	0.0275	0.0143	0.0657	0.0504
SAMME	0.0844	0.1284	0.0369	0.0196	0.0636	0.0726
SAMME.R	0.6503	0.2506	0.2207	0.2123	0.1477	0.7317
RobAdaBoost	0.0682	0.0965	0.0296	0.0139	0.0625	0.0484
ASSEMBLE	0.0813	0.0817	0.0221	0.0154	0.0723	0.0525
SemiBoost	0.0856	0.1510	0.0193	0.0121	0.0808	0.0609
Baseline	0.0671	0.0994	0.0289	0.0119	0.0695	0.0544
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0619	0.0067	0.1344	0.0317	0.2212	0.0659
FWAdaBoost	0.0603	0.0083	0.1491	0.0411	0.2227	0.0739
AdaBoost.M1	0.0600	0.0115	0.1724	0.0369	0.2114	0.0683
AdaBoost.M2	0.0619	0.0086	0.1677	0.0352	0.2529	0.0704
SAMME	0.0627	0.0126	0.2062	0.0522	0.2114	0.0754
SAMME.R	0.1009	0.0424	0.6210	0.1858	0.3530	0.4320
RobAdaBoost	0.0580	0.0082	0.1495	0.0340	0.2145	0.0649
ASSEMBLE	0.0616	0.0059	0.1480	0.0297	0.2405	0.0759
SemiBoost	0.1082	0.0957	0.1551	0.1487	0.4362	0.0776
Baseline	0.0601	0.0071	0.1538	0.0307	0.1995	0.0585

Table S17. Average ranks of classification performances of SSFWAdaBoost and alternative boosting algorithms over 12 benchmark problems

EFS	**Algorithm**	**Splitting Ratio**			
		1:19	**1:9**	**3:17**	**1:4**
SOFIS	SSFWAdaBoost	**2.5833**	**2.2500**	**2.1667**	**2.0000**
	FWAdaBoost	5.0417	4.5417	4.5000	4.5417
	AdaBoost.M1	6.8333	7.1250	7.0417	7.3750
	AdaBoost.M2	7.5833	7.3750	7.1667	7.4167
	SAMME	8.3750	8.7083	8.5417	8.2500
	SAMME.R	7.2500	6.4167	6.0833	6.2917
	RobAdaBoost	3.0833	3.0000	3.0000	2.9583
	ASSEMBLE	3.0833	3.1667	3.0833	2.7500
	SemiBoost	5.3333	5.2500	5.8333	6.0000
	Baseline	5.8333	7.1667	7.5833	7.4167
ALMMo0	SSFWAdaBoost	**2.5833**	**2.4167**	**2.0833**	**1.9167**
	FWAdaBoost	4.7500	4.4167	4.3333	4.2917
	AdaBoost.M1	7.1667	7.0000	7.3333	7.7083
	AdaBoost.M2	8.0000	7.6667	7.3750	7.1667
	SAMME	8.3333	8.2500	8.7500	8.5417
	SAMME.R	6.5000	5.3333	5.1667	4.8333
	RobAdaBoost	3.6667	3.4167	3.4167	3.0417
	ASSEMBLE	3.0000	3.3750	3.7500	3.6667
	SemiBoost	5.5000	5.5000	5.5000	5.8333
	Baseline	5.5000	7.6250	7.2917	8.0000
SOFBIS	SSFWAdaBoost	**2.0833**	**2.1667**	**2.6250**	**3.2917**
	FWAdaBoost	5.5833	5.5000	5.2500	5.5833
	AdaBoost.M1	5.6667	5.5000	5.0833	4.6250
	AdaBoost.M2	7.1667	5.7917	5.5833	5.5417
	SAMME	7.5833	7.4167	7.2500	7.4583
	SAMME.R	9.8333	9.8333	9.8333	9.7500
	RobAdaBoost	3.6667	3.2083	3.0833	3.2500
	ASSEMBLE	4.7500	4.8333	5.2083	4.4167
	SemiBoost	6.3333	7.1667	7.2500	7.6667
	Baseline	2.3333	3.5833	3.8333	3.4167

Table S18. Performance comparison between SSFWAdaBoost and alternative boosting algorithms over 12 benchmark problems in terms of classification error rate on labelled training samples

EFS	**Algorithm**	**Splitting Ratio**			
		1:19	**1:9**	**3:17**	**1:4**
SOFIS	SSFWAdaBoost	0.0011	0.0009	0.0007	0.0004
	FWAdaBoost	0.0018	0.0012	0.0013	0.0008
	AdaBoost.M1	0.0798	0.0642	0.0568	0.0500
	AdaBoost.M2	0.1635	0.1204	0.0908	0.0781
	SAMME	0.1197	0.0840	0.0689	0.0588
	SAMME.R	0.1670	0.0924	0.0652	0.0574
	RobAdaBoost	0.0528	0.0425	0.0352	0.0316
	ASSEMBLE	0.0292	0.0448	0.0502	0.0490
	SemiBoost	0.0514	0.0459	0.0402	0.0352
	Baseline	0.0848	0.0698	0.0591	0.0491
ALMMo0	SSFWAdaBoost	0.0001	0.0001	0.0000	0.0000
	FWAdaBoost	0.0006	0.0006	0.0006	0.0005
	AdaBoost.M1	0.0636	0.0629	0.0578	0.0533
	AdaBoost.M2	0.1319	0.1060	0.0857	0.0762
	SAMME	0.1268	0.1026	0.0934	0.0867
	SAMME.R	0.1501	0.0761	0.0455	0.0379
	RobAdaBoost	0.0368	0.0364	0.0327	0.0307
	ASSEMBLE	0.0414	0.0528	0.0507	0.0495
	SemiBoost	0.0928	0.0873	0.0799	0.0749
	Baseline	0.0676	0.0673	0.0562	0.0562
SOFBIS	SSFWAdaBoost	0.0035	0.0035	0.0039	0.0042
	FWAdaBoost	0.0010	0.0017	0.0014	0.0018
	AdaBoost.M1	0.0231	0.0217	0.0189	0.0179
	AdaBoost.M2	0.0693	0.0407	0.0310	0.0260
	SAMME	0.0676	0.0516	0.0410	0.0368
	SAMME.R	0.3691	0.3808	0.3514	0.3013
	RobAdaBoost	0.0093	0.0109	0.0103	0.0112
	ASSEMBLE	0.0375	0.0383	0.0337	0.0333
	SemiBoost	0.0311	0.0319	0.0306	0.0280
	Baseline	0.0092	0.0115	0.0107	0.0108

Table S19. *p*-values returned by Friedman tests for evaluating the statistical significance of SSFWAdaBoost over alternative boosting algorithms

EFS	**SOFIS**	**ALMMo0**	**SOFBIS**
p-value	0.0000	0.0000	0.0000

Table S20. *p*-values returned by pairwise Wilcoxon signed rank tests for evaluating the statistical significance of SSFWAdaBoost over alternative boosting algorithms

SSFWAdaBoost versus	**EFS**		
	SOFIS	**ALMMo0**	**SOFBIS**
FWAdaBoost	0.0000	0.0000	0.0000
AdaBoost.M1	0.0000	0.0000	0.0000
AdaBoost.M2	0.0000	0.0000	0.0000
SAMME	0.0000	0.0000	0.0000
SAMME.R	0.0000	0.0000	0.0000
RobAdaBoost	0.0000	0.0000	0.0009
ASSEMBLE	0.8535	0.0003	0.0000
SemiBoost	0.0056	0.0012	0.0000
Baseline	0.0000	0.0000	0.0053

Table S21. Performance comparison between SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems under splitting ratio 1:19

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.2800	0.2207	0.0341	0.0349	0.1247	0.1441
SSFWAdaBoost-ALMMo0	0.2603	0.2036	0.0372	0.0315	0.1421	0.1208
SSFWAdaBoost-SOFBIS	0.1263	0.1988	0.0366	0.0332	0.1005	0.0671
LGC	0.1504	0.1842	0.0362	0.0273	0.1536	0.0251
GGMC	0.2455	0.5036	0.0751	0.0960	0.3465	0.0233
AGRK	0.1972	0.1977	0.0239	0.0221	0.1366	0.1434
AGRL	0.1932	0.2121	0.0252	0.0237	0.1345	0.1541
EAGR	0.1965	0.1937	0.0202	0.0274	0.1314	0.1329
LSVM	0.2176	0.2051	0.0263	0.0638	0.2004	0.1389
TMPM	0.2500	0.5074	0.1190	0.1443	0.0846	0.1037
STHP	0.2723	0.2056	0.0298	0.0337	0.1352	0.1429
TriTrain-DT	0.1236	0.3839	0.2550	0.1537	0.0904	0.2087
TriTrain-kNN	0.2226	0.2825	0.0421	0.0429	0.1282	0.1775
ASSEMBLE-DT	0.1685	0.2366	0.0653	0.0452	0.1015	0.1611
ASSEMBLE-kNN	0.1836	0.2117	0.0350	0.0391	0.1195	0.1589
SemiBoost-DT	0.1503	0.2671	0.0569	0.0453	0.0801	0.1854
SemiBoost-kNN	0.1822	0.2652	0.0348	0.0354	0.1042	0.2392
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0526	0.0245	0.2962	0.0437	0.3744	0.2760
SSFWAdaBoost-ALMMo0	0.0562	0.0229	0.2685	0.0337	0.3539	0.2500
SSFWAdaBoost-SOFBIS	0.0789	0.0253	0.2311	0.0625	0.3403	0.1294
LGC	0.0589	0.0223	0.1921	0.0374	0.3392	0.1409
GGMC	0.6711	0.0267	0.2472	0.0378	0.6153	0.2551
AGRK	0.2757	0.0310	0.2364	0.0468	0.3984	0.1962
AGRL	0.2721	0.0395	0.2395	0.0540	0.4149	0.1987
EAGR	0.3178	0.0923	0.1861	0.0471	0.4123	0.1913
LSVM	0.1013	0.0211	0.2073	0.1232	0.4182	0.2110
TMPM	0.1223	0.0277	0.2824	0.0521	0.5308	0.2393
STHP	0.0916	0.0268	0.2344	0.0429	0.4315	0.2588
TriTrain-DT	0.0608	0.0236	0.5417	0.2127	0.4128	0.1360
TriTrain-kNN	0.0750	0.0326	0.3258	0.0728	0.4042	0.2166
ASSEMBLE-DT	0.0664	0.0201	0.3135	0.2428	0.4894	0.1543
ASSEMBLE-kNN	0.0835	0.0380	0.2622	0.0623	0.3453	0.1931
SemiBoost-DT	0.1503	0.1055	0.2548	0.0819	0.3474	0.1636
SemiBoost-kNN	0.1769	0.1051	0.2474	0.2628	0.4995	0.1931

Table S22. Performance comparison between SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems under splitting ratio 1:9

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.2119	0.1474	0.0277	0.0206	0.1012	0.1011
SSFWAdaBoost-ALMMo0	0.1963	0.1357	0.0286	0.0177	0.1116	0.0903
SSFWAdaBoost-SOFBIS	0.0981	0.1359	0.0299	0.0213	0.0818	0.0557
LGC	0.1155	0.1437	0.0354	0.0202	0.1256	0.0222
GGMC	0.2325	0.4181	0.0501	0.0814	0.2869	0.0229
AGRK	0.1795	0.1560	0.0251	0.0164	0.1209	0.1233
AGRL	0.1869	0.1740	0.0261	0.0185	0.1202	0.1483
EAGR	0.1752	0.1452	0.0197	0.0172	0.1074	0.1140
LSVM	0.1624	0.1639	0.0234	0.0351	0.1374	0.1010
TMPM	0.1856	0.4702	0.0867	0.1170	0.0784	0.0691
STHP	0.2037	0.1382	0.0234	0.0221	0.1096	0.1005
TriTrain-DT	0.0995	0.3119	0.1918	0.0430	0.0795	0.1064
TriTrain-kNN	0.1602	0.1918	0.0285	0.0234	0.1087	0.1252
ASSEMBLE-DT	0.0988	0.1664	0.0483	0.0318	0.0741	0.0797
ASSEMBLE-kNN	0.1503	0.1336	0.0246	0.0219	0.1079	0.1232
SemiBoost-DT	0.0974	0.2089	0.0448	0.0251	0.0919	0.1408
SemiBoost-kNN	0.1395	0.2114	0.0256	0.0192	0.0954	0.1511
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0447	0.0157	0.1813	0.0268	0.3077	0.2128
SSFWAdaBoost-ALMMo0	0.0466	0.0155	0.1950	0.0222	0.2865	0.1926
SSFWAdaBoost-SOFBIS	0.0640	0.0142	0.1715	0.0439	0.2758	0.0978
LGC	0.0535	0.0195	0.1763	0.0298	0.2850	0.1147
GGMC	0.6420	0.0232	0.2220	0.0372	0.5256	0.2052
AGRK	0.3076	0.0137	0.2080	0.0402	0.3414	0.1709
AGRL	0.2871	0.0234	0.2105	0.0461	0.3727	0.1774
EAGR	0.3224	0.0702	0.1602	0.0394	0.3609	0.1686
LSVM	0.0959	0.0088	0.1658	0.1262	0.4724	0.1598
TMPM	0.1257	0.0262	0.2301	0.0306	0.5138	0.1992
STHP	0.0906	0.0190	0.1740	0.0346	0.3992	0.2000
TriTrain-DT	0.0447	0.0130	0.4526	0.1564	0.4672	0.1034
TriTrain-kNN	0.0613	0.0162	0.2283	0.0465	0.3894	0.1595
ASSEMBLE-DT	0.0516	0.0106	0.2392	0.0671	0.2714	0.0914
ASSEMBLE-kNN	0.0666	0.0134	0.1836	0.0399	0.2830	0.1466
SemiBoost-DT	0.1169	0.1042	0.1971	0.2117	0.4679	0.0959
SemiBoost-kNN	0.1412	0.1039	0.1918	0.2371	0.4883	0.1370

Table S23. Performance comparison between SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems under splitting ratio 3:17

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.1754	0.1117	0.0218	0.0159	0.0903	0.0861
SSFWAdaBoost-ALMMo0	0.1569	0.1036	0.0232	0.0147	0.0971	0.0803
SSFWAdaBoost-SOFBIS	0.0817	0.1047	0.0246	0.0192	0.0707	0.0444
LGC	0.1158	0.1289	0.0329	0.0195	0.1174	0.0205
GGMC	0.1796	0.3636	0.0414	0.0727	0.2496	0.0238
AGRK	0.1577	0.1324	0.0233	0.0156	0.1060	0.1112
AGRL	0.1593	0.1519	0.0236	0.0166	0.1072	0.1354
EAGR	0.1630	0.1211	0.0172	0.0151	0.0936	0.1071
LSVM	0.1405	0.1437	0.0200	0.0177	0.1177	0.0931
TMPM	0.1628	0.4614	0.0767	0.1147	0.0778	0.0629
STHP	0.1609	0.1077	0.0215	0.0177	0.0951	0.0818
TriTrain-DT	0.0840	0.2734	0.1716	0.0955	0.0734	0.0922
TriTrain-kNN	0.1347	0.1477	0.0246	0.0178	0.0999	0.1059
ASSEMBLE-DT	0.0804	0.1367	0.0445	0.0274	0.0648	0.0581
ASSEMBLE-kNN	0.1187	0.1051	0.0210	0.0178	0.0953	0.1028
SemiBoost-DT	0.0762	0.1773	0.0385	0.0200	0.0856	0.1126
SemiBoost-kNN	0.1138	0.1758	0.0228	0.0145	0.0889	0.1346
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0435	0.0138	0.1397	0.0207	0.2705	0.1686
SSFWAdaBoost-ALMMo0	0.0454	0.0130	0.1629	0.0172	0.2540	0.1523
SSFWAdaBoost-SOFBIS	0.0643	0.0103	0.1425	0.0346	0.2438	0.0744
LGC	0.0518	0.0206	0.1670	0.0278	0.2634	0.1090
GGMC	0.6575	0.0227	0.2120	0.0323	0.4700	0.1582
AGRK	0.3280	0.0123	0.1842	0.0371	0.3172	0.1523
AGRL	0.2839	0.0188	0.1851	0.0445	0.3511	0.1574
EAGR	0.3088	0.0537	0.1458	0.0354	0.3399	0.1547
LSVM	0.0921	0.0055	0.1518	0.0941	0.3622	0.1260
TMPM	0.1241	0.0264	0.1914	0.0216	0.5398	0.1432
STHP	0.1136	0.0133	0.1420	0.0276	0.3758	0.1535
TriTrain-DT	0.0439	0.0104	0.4027	0.1416	0.3092	0.0809
TriTrain-kNN	0.0577	0.0110	0.2018	0.0394	0.2875	0.1287
ASSEMBLE-DT	0.0465	0.0080	0.2145	0.0542	0.2303	0.0688
ASSEMBLE-kNN	0.0619	0.0097	0.1518	0.0310	0.2526	0.1166
SemiBoost-DT	0.0947	0.1031	0.1639	0.1923	0.4372	0.0717
SemiBoost-kNN	0.1177	0.1004	0.1631	0.2188	0.4608	0.1079

Table S24. Performance comparison between SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems under splitting ratio 1:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.1541	0.0924	0.0197	0.0123	0.0791	0.0731
SSFWAdaBoost-ALMMo0	0.1377	0.0872	0.0211	0.0116	0.0826	0.0683
SSFWAdaBoost-SOFBIS	0.0702	0.0852	0.0240	0.0146	0.0636	0.0446
LGC	0.1056	0.1192	0.0322	0.0182	0.1082	0.0201
GGMC	0.1490	0.3062	0.0414	0.0563	0.2170	0.0234
AGRK	0.1540	0.1169	0.0223	0.0124	0.0966	0.1043
AGRL	0.1586	0.1367	0.0235	0.0136	0.0955	0.1234
EAGR	0.1558	0.1074	0.0178	0.0120	0.0897	0.0999
LSVM	0.1293	0.1175	0.0204	0.0166	0.1087	0.0849
TMPM	0.1290	0.4636	0.0706	0.1098	0.0787	0.0466
STHP	0.1411	0.0911	0.0197	0.0134	0.0854	0.0738
TriTrain-DT	0.0753	0.2388	0.1504	0.0816	0.0697	0.0856
TriTrain-kNN	0.1180	0.1190	0.0215	0.0138	0.0884	0.0922
ASSEMBLE-DT	0.0583	0.1184	0.0409	0.0233	0.0599	0.0488
ASSEMBLE-kNN	0.1106	0.0868	0.0178	0.0140	0.0866	0.0879
SemiBoost-DT	0.0622	0.1582	0.0362	0.0162	0.0787	0.0972
SemiBoost-kNN	0.1028	0.1579	0.0188	0.0110	0.0841	0.1080
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0410	0.0106	0.1240	0.0172	0.2429	0.1460
SSFWAdaBoost-ALMMo0	0.0441	0.0103	0.1474	0.0142	0.2367	0.1253
SSFWAdaBoost-SOFBIS	0.0619	0.0067	0.1344	0.0317	0.2212	0.0659
LGC	0.0496	0.0177	0.1609	0.0262	0.2562	0.0974
GGMC	0.6351	0.0192	0.2042	0.0315	0.4256	0.1532
AGRK	0.3160	0.0097	0.1695	0.0322	0.2984	0.1465
AGRL	0.2855	0.0156	0.1744	0.0376	0.3345	0.1555
EAGR	0.3008	0.0401	0.1467	0.0330	0.3229	0.1546
LSVM	0.0900	0.0054	0.1515	0.0866	0.3516	0.1187
TMPM	0.1132	0.0262	0.1860	0.0180	0.5084	0.1375
STHP	0.1022	0.0110	0.1361	0.0252	0.3665	0.1313
TriTrain-DT	0.0424	0.0066	0.3720	0.1309	0.2816	0.0774
TriTrain-kNN	0.0542	0.0065	0.1859	0.0346	0.2601	0.1054
ASSEMBLE-DT	0.0453	0.0046	0.1950	0.0490	0.2000	0.0625
ASSEMBLE-kNN	0.0574	0.0050	0.1450	0.0265	0.2302	0.1032
SemiBoost-DT	0.0814	0.1009	0.1565	0.1718	0.4117	0.0620
SemiBoost-kNN	0.1060	0.0960	0.1560	0.1980	0.4384	0.0886

Table S25. Average ranks of classification performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems

Algorithm	**Splitting Ratio**			
	1:19	**1:9**	**3:17**	**1:4**
SSFWAdaBoost-SOFIS	9.0000	7.8750	6.9167	6.0417
SSFWAdaBoost-ALMMo0	7.7500	7.0833	5.9583	5.6667
SSFWAdaBoost-SOFBIS	5.0833	**5.1667**	**4.9583**	**4.9167**
LGC	**4.0000**	6.4167	8.0000	8.5000
GGMC	12.0833	13.6667	13.7500	13.4167
AGRK	7.6667	9.1667	10.2083	10.8333
AGRL	9.0833	11.2500	12.0833	12.7500
EAGR	7.1667	7.9167	9.4167	10.2917
LSVM	9.1667	9.0417	8.6667	9.6667
TMPM	11.6667	11.9167	11.9167	11.7083
STHP	9.2500	8.6250	8.0417	7.9583
TriTrain-DT	10.1667	9.7083	9.5833	9.6667
TriTrain-kNN	11.5000	10.2500	9.6667	8.8333
ASSEMBLE-DT	9.5833	6.9167	6.8333	6.6667
ASSEMBLE-kNN	8.7083	6.5833	6.2500	5.7917
SemiBoost-DT	9.9167	10.6667	10.7500	10.2917
SemiBoost-kNN	11.2083	10.7500	10.0000	10.0000

Table S26. Performance demonstration of SSFWAdaBoost over 12 benchmark problems under splitting ratio 1:19 with $\frac{1}{3}$ of pseudo-labelling errors made by the initial base classifier $h_0(\boldsymbol{x})$ randomly corrected for training the remaining ensemble components

EFS	**Algorithm**	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SOFBIS	SSFWAdaBoost$^{1/3}$	0.0890	0.1481	0.0353	0.0244	0.0706	0.0590
	SSFWAdaBoost	0.1263	0.1988	0.0366	0.0332	0.1005	0.0671
ALMMo0	SSFWAdaBoost$^{1/3}$	0.2145	0.1913	0.0346	0.0306	0.0922	0.1044
	SSFWAdaBoost	0.2603	0.2036	0.0372	0.0315	0.1421	0.1208
SOFIS	SSFWAdaBoost$^{1/3}$	0.2471	0.1961	0.0379	0.0343	0.0865	0.1234
	SSFWAdaBoost	0.2800	0.2207	0.0341	0.0349	0.1247	0.1441
	Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SOFBIS	SSFWAdaBoost$^{1/3}$	0.0533	0.0168	0.1767	0.0525	0.2242	0.0885
	SSFWAdaBoost	0.0789	0.0253	0.2311	0.0625	0.3403	0.1294
ALMMo0	SSFWAdaBoost$^{1/3}$	0.0417	0.0204	0.1927	0.0372	0.3091	0.2179
	SSFWAdaBoost	0.0562	0.0229	0.2685	0.0337	0.3539	0.2500
SOFIS	SSFWAdaBoost$^{1/3}$	0.0394	0.0194	0.1822	0.0378	0.2852	0.2354
	SSFWAdaBoost	0.0526	0.0245	0.2962	0.0437	0.3744	0.2760

Table S27. *p*-values returned by pairwise Wilcoxon signed rank tests for evaluating the statistical significance of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS over alternative semi-supervised classification models

SSFWAdaBoost	**EFS**		
versus	**SOFIS**	**ALMMo0**	**SOFBIS**
LGC	0.2614	0.5451	0.0564
GGMC	0.0001	0.0000	0.0000
AGRK	0.0231	0.0012	0.0000
AGRL	0.0010	0.0000	0.0000
EAGR	0.0472	0.0107	0.0000
LSVM	0.0489	0.0107	0.0000
TMPM	0.0012	0.0001	0.0000
STHP	0.5117	0.0005	0.0000
TriTrain-DT	0.0027	0.0020	0.0000
TriTrain-kNN	0.0089	0.0014	0.0000
ASSEMBLE-DT	0.7273	0.8980	0.0015
ASSEMBLE-kNN	0.4540	0.6666	0.0000
SemiBoost-DT	0.0289	0.0125	0.0000
SemiBoost-kNN	0.0035	0.0016	0.0000

Table S28. Performance comparison between different semi-supervised ensemble classifiers over four high-dimensional problems under two different splitting ratios.

Splitting Ratio	**Algorithm**	**MNIST**	**FMNIST**	**PMNIST**	**RMNIST**
1:19	SSFWAdaBoost-SOFIS	0.1275	0.2488	0.1057	0.1033
	SSFWAdaBoost-ALMMo0	0.1246	0.2670	0.1108	0.1078
	SSFWAdaBoost-SOFBIS	0.1373	0.2685	0.1363	0.1351
	TriTrain-DT	0.3748	0.3473	0.3810	0.3794
	TriTrain-kNN	0.1631	0.2739	0.1652	0.1601
	TriTrain-SOFIS	0.1088	0.2331	0.0944	0.0963
	TriTrain-ALMMo0	0.1439	0.2638	0.1252	0.1254
	TriTrain-SOFBIS	0.1508	0.2591	0.1527	0.1525
	ASSEMBLE-DT	0.1762	0.2823	0.1775	0.1782
	ASSEMBLE-kNN	0.1199	0.2765	0.1204	0.1255
	ASSEMBLE-SOFIS	0.1271	0.2601	0.1136	0.1162
	ASSEMBLE-ALMMo0	0.1306	0.2740	0.1530	0.1553
	ASSEMBLE-SOFBIS	0.1471	0.3022	0.1504	0.1504
	SemiBoost-DT	0.2547	0.2842	0.2589	0.2541
	SemiBoost-kNN	0.2130	0.2894	0.2151	0.2122
	SemiBoost-SOFIS	0.1945	0.2638	0.1771	0.1763
	SemiBoost-ALMMo0	0.2035	0.2720	0.1824	0.1786
	SemiBoost-SOFBIS	0.2071	0.2749	0.2112	0.2086
1:9	SSFWAdaBoost-SOFIS	0.0971	0.2135	0.0725	0.0733
	SSFWAdaBoost-ALMMo0	0.0980	0.2408	0.0870	0.0847
	SSFWAdaBoost-SOFBIS	0.1094	0.2481	0.1046	0.1079
	TriTrain-DT	0.3186	0.3188	0.3075	0.3144
	TriTrain-kNN	0.1239	0.2490	0.1232	0.1255
	TriTrain-SOFIS	0.0868	0.2083	0.0734	0.0722
	TriTrain-ALMMo0	0.1156	0.2419	0.0982	0.1000
	TriTrain-SOFBIS	0.1182	0.2403	0.1175	0.1182
	ASSEMBLE-DT	0.1447	0.2614	0.1459	0.1472
	ASSEMBLE-kNN	0.0966	0.2529	0.0949	0.0980
	ASSEMBLE-SOFIS	0.0919	0.2230	0.0774	0.0785
	ASSEMBLE-ALMMo0	0.0966	0.2436	0.1170	0.1177
	ASSEMBLE-SOFBIS	0.1075	0.2860	0.1085	0.1111
	SemiBoost-DT	0.2202	0.2650	0.2169	0.2150
	SemiBoost-kNN	0.1809	0.2703	0.1830	0.1798
	SemiBoost-SOFIS	0.1572	0.2408	0.1364	0.1364
	SemiBoost-ALMMo0	0.1696	0.2522	0.1505	0.1492
	SemiBoost-SOFBIS	0.1738	0.2555	0.1752	0.1733

Table S29. Average ranks of different semi-supervised ensemble classifiers over four high-dimensional problems under two different splitting ratios.

Algorithm	**Splitting Ratio**	
	1:19	**1:9**
SSFWAdaBoost-SOFIS	2.7500	2.5000
SSFWAdaBoost-ALMMo0	4.0000	4.8750
SSFWAdaBoost-SOFBIS	7.2500	7.7500
TriTrain-DT	18.0000	18.0000
TriTrain-kNN	10.7500	10.7500
TriTrain-SOFIS	**1.0000**	**1.2500**
TriTrain-ALMMo0	6.1250	7.0000
TriTrain-SOFBIS	7.7500	8.5000
ASSEMBLE-DT	13.0000	13.0000
ASSEMBLE-kNN	6.5000	6.3750
ASSEMBLE-SOFIS	4.0000	2.7500
ASSEMBLE-ALMMo0	9.2500	7.3750
ASSEMBLE-SOFBIS	10.5000	10.0000
SemiBoost-DT	16.5000	16.5000
SemiBoost-kNN	16.0000	16.0000
SemiBoost-SOFIS	10.6250	10.6250
SemiBoost-ALMMo0	12.7500	13.2500
SemiBoost-SOFBIS	14.2500	14.5000

Table S30. *p*-values returned by pairwise Wilcoxon signed rank tests for evaluating the statistical significance of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS over alternative semi-supervised ensemble classification models on high-dimensional problems

SSFWAdaBoost	**EFS**		
versus	**SOFIS**	**ALMMo0**	**SOFBIS**
TriTrain-DT	0.0078	0.0078	0.0078
TriTrain-kNN	0.0078	0.0078	0.0078
TriTrain-SOFIS	0.0156	0.0078	0.0078
TriTrain-ALMMo0	0.0078	0.0234	0.1641
TriTrain-SOFBIS	0.0078	0.0391	0.0547
ASSEMBLE-DT	0.0078	0.0078	0.0078
ASSEMBLE-kNN	0.0391	0.0391	0.0391
ASSEMBLE-SOFIS	0.0625	0.2500	0.0078
ASSEMBLE-ALMMo0	0.0156	0.0156	0.3125
ASSEMBLE-SOFBIS	0.0078	0.0078	0.0156
SemiBoost-DT	0.0078	0.0078	0.0078
SemiBoost-kNN	0.0078	0.0078	0.0078
SemiBoost-SOFIS	0.0078	0.0312	0.0391
SemiBoost-ALMMo0	0.0078	0.0078	0.0078
SemiBoost-SOFBIS	0.0078	0.0078	0.0078

Table S31. Performance demonstration of SSFWAdaBoost over 12 benchmark problems under splitting ratio 1:19 with 5% mislabelled training samples

EFS	**Algorithm**	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SOFIS	SSFWAdaBoost	0.2912	0.2462	0.0646	0.0590	0.1546	0.1570
	Baseline	0.3742	0.3305	0.0800	0.0735	0.1571	0.2117
ALMMo0	SSFWAdaBoost	0.2860	0.2443	0.0848	0.0777	0.1781	0.1482
	Baseline	0.3573	0.3360	0.1144	0.1155	0.1794	0.1921
SOFBIS	SSFWAdaBoost	0.1639	0.2252	0.0439	0.0456	0.1263	0.0957
	Baseline	0.1738	0.2504	0.0672	0.0604	0.1358	0.1169
EFS	**Algorithm**	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SOFIS	SSFWAdaBoost	0.0576	0.0411	0.3006	0.0673	0.3861	0.3110
	Baseline	0.0870	0.0539	0.2931	0.1017	0.4395	0.3917
ALMMo0	SSFWAdaBoost	0.0949	0.0575	0.3058	0.0831	0.3803	0.2985
	Baseline	0.1561	0.0827	0.3272	0.1154	0.4746	0.3690
SOFBIS	SSFWAdaBoost	0.0941	0.0546	0.2507	0.0776	0.3599	0.1751
	Baseline	0.1028	0.0579	0.2913	0.1018	0.3584	0.1815

Table S32. Performance demonstration of alternative EFSs boosted by SSFWAdaBoost over 12 benchmark problems

EFS	**Algorithm**	**Splitting Ratio**			
		1:19	**1:9**	**3:17**	**1:4**
eClass0	SSFWAdaBoost	0.2539	0.2441	0.2309	0.2305
	FWAdaBoost	0.2228	0.2128	0.2012	0.1998
	Baseline	0.3386	0.3306	0.3280	0.3196
ALMMo1	SSFWAdaBoost	0.1873	0.1865	0.1776	0.1683
	FWAdaBoost	0.1753	0.1629	0.1571	0.1552
	Baseline	0.2013	0.2082	0.2004	0.1919
SAFL	SSFWAdaBoost	0.1119	0.0952	0.0902	0.0877
	FWAdaBoost	0.1334	0.1053	0.0937	0.0893
	Baseline	0.1246	0.1029	0.0958	0.0927

Table S33. Performance demonstration of SSFWAdaBoost with eClass0 as base classifier over 12 benchmark problems

Splitting Ratio	**Algorithm**	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
1:19	SSFWAdaBoost	0.2466	0.4009	0.0795	0.1320	0.1747	0.2043
	FWAdaBoost	0.2420	0.4128	0.0857	0.1213	0.1483	0.2114
	Baseline	0.3187	0.5600	0.1446	0.1646	0.2119	0.2563
1:9	SSFWAdaBoost	0.2456	0.3621	0.0711	0.1273	0.1744	0.1893
	FWAdaBoost	0.2309	0.3614	0.0765	0.1197	0.1483	0.1868
	Baseline	0.3194	0.5416	0.1190	0.1667	0.2252	0.2416
3:17	SSFWAdaBoost	0.2221	0.3417	0.0696	0.1244	0.1749	0.1910
	FWAdaBoost	0.2174	0.3386	0.0704	0.1085	0.1537	0.1773
	Baseline	0.3012	0.5322	0.1310	0.1735	0.2175	0.2252
1:4	SSFWAdaBoost	0.2196	0.3358	0.0679	0.1211	0.1649	0.1674
	FWAdaBoost	0.2070	0.3335	0.0679	0.1034	0.1518	0.1774
	Baseline	0.2835	0.5251	0.1223	0.1576	0.2357	0.2226
Splitting Ratio	**Algorithm**	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
1:19	SSFWAdaBoost	0.3994	0.0992	0.3200	0.1675	0.5691	0.2532
	FWAdaBoost	0.1598	0.0725	0.3167	0.1666	0.5000	0.2368
	Baseline	0.4764	0.2941	0.4020	0.2234	0.6830	0.3286
1:9	SSFWAdaBoost	0.4402	0.1018	0.2524	0.1802	0.5529	0.2325
	FWAdaBoost	0.2104	0.0752	0.2608	0.1640	0.4913	0.2281
	Baseline	0.5114	0.2803	0.3508	0.2274	0.6727	0.3111
3:17	SSFWAdaBoost	0.4293	0.0745	0.2209	0.1655	0.5367	0.2204
	FWAdaBoost	0.2104	0.0529	0.2307	0.1560	0.4937	0.2053
	Baseline	0.4862	0.3401	0.3439	0.2099	0.6786	0.2967
1:4	SSFWAdaBoost	0.4411	0.1053	0.2177	0.1728	0.5441	0.2088
	FWAdaBoost	0.2023	0.0655	0.2191	0.1600	0.4908	0.2196
	Baseline	0.4762	0.2857	0.3352	0.2112	0.6916	0.2884

Table S34. Performance demonstration of SSFWAdaBoost with ALMMo1 as base classifier over 12 benchmark problems

Splitting Ratio	**Algorithm**	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
1:19	SSFWAdaBoost	0.1801	0.4542	0.1050	0.1482	0.0839	0.0501
	FWAdaBoost	0.1862	0.4511	0.1213	0.1523	0.0837	0.0604
	Baseline	0.2005	0.4782	0.1070	0.1468	0.0837	0.0848
1:9	SSFWAdaBoost	0.1644	0.4436	0.0851	0.1347	0.0820	0.0426
	FWAdaBoost	0.1666	0.4383	0.0918	0.1355	0.0806	0.0464
	Baseline	0.1977	0.4832	0.0864	0.1398	0.0826	0.0714
3:17	SSFWAdaBoost	0.1556	0.4383	0.0780	0.1335	0.0806	0.0463
	FWAdaBoost	0.1526	0.4379	0.0843	0.1361	0.0794	0.0326
	Baseline	0.1883	0.4575	0.0872	0.1396	0.0807	0.0916
1:4	SSFWAdaBoost	0.1473	0.4441	0.0744	0.1305	0.0824	0.0431
	FWAdaBoost	0.1423	0.4353	0.0800	0.1309	0.0803	0.0282
	Baseline	0.1760	0.4915	0.0771	0.1433	0.0823	0.1174
Splitting Ratio	**Algorithm**	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
1:19	SSFWAdaBoost	0.0761	0.0266	0.4854	0.0201	0.4468	0.1708
	FWAdaBoost	0.0745	0.0264	0.2983	0.0337	0.4465	0.1696
	Baseline	0.0824	0.0269	0.5034	0.0197	0.4530	0.2291
1:9	SSFWAdaBoost	0.0751	0.0272	0.5619	0.0149	0.4411	0.1651
	FWAdaBoost	0.0718	0.0272	0.2648	0.0267	0.4391	0.1665
	Baseline	0.0806	0.0268	0.6591	0.0152	0.4732	0.1823
3:17	SSFWAdaBoost	0.0749	0.0238	0.4883	0.0144	0.4394	0.1582
	FWAdaBoost	0.0709	0.0241	0.2518	0.0207	0.4363	0.1591
	Baseline	0.0783	0.0236	0.5858	0.0148	0.4674	0.1899
1:4	SSFWAdaBoost	0.0761	0.0202	0.3962	0.0133	0.4392	0.1532
	FWAdaBoost	0.0746	0.0222	0.2622	0.0183	0.4322	0.1559
	Baseline	0.0799	0.0205	0.4698	0.0132	0.4547	0.1774

Table S35. Performance demonstration of SSFWAdaBoost with SAFL as base classifier over 12 benchmark problems

Splitting Ratio	Algorithm	SE	LR	OR	PR	PW	MF
1:19	SSFWAdaBoost	0.1176	0.2416	0.0342	0.0286	0.0801	0.0578
	FWAdaBoost	0.1651	0.2582	0.0478	0.0378	0.0812	0.0882
	Baseline	0.1386	0.2450	0.0467	0.0308	0.0802	0.0976
1:9	SSFWAdaBoost	0.0993	0.2202	0.0251	0.0204	0.0726	0.0394
	FWAdaBoost	0.1177	0.2215	0.0341	0.0243	0.0728	0.0496
	Baseline	0.1065	0.2222	0.0312	0.0215	0.0730	0.0645
3:17	SSFWAdaBoost	0.0993	0.2122	0.0208	0.0180	0.0694	0.0329
	FWAdaBoost	0.1031	0.2092	0.0264	0.0216	0.0685	0.0368
	Baseline	0.1066	0.2163	0.0236	0.0192	0.0700	0.0506
1:4	SSFWAdaBoost	0.0962	0.2082	0.0188	0.0154	0.0706	0.0269
	FWAdaBoost	0.1015	0.2033	0.0243	0.0169	0.0692	0.0315
	Baseline	0.1030	0.2126	0.0215	0.0158	0.0703	0.0376
Splitting Ratio	**Algorithm**	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
1:19	SSFWAdaBoost	0.0742	0.0140	0.2057	0.0100	0.3652	0.1136
	FWAdaBoost	0.0750	0.0157	0.2790	0.0245	0.3740	0.1543
	Baseline	0.0746	0.0140	0.2408	0.0203	0.3723	0.1348
1:9	SSFWAdaBoost	0.0773	0.0105	0.1387	0.0059	0.3349	0.0976
	FWAdaBoost	0.0630	0.0110	0.2038	0.0112	0.3390	0.1153
	Baseline	0.0770	0.0109	0.1682	0.0112	0.3433	0.1057
3:17	SSFWAdaBoost	0.0830	0.0081	0.1099	0.0053	0.3264	0.0970
	FWAdaBoost	0.0654	0.0104	0.1463	0.0091	0.3234	0.1045
	Baseline	0.0827	0.0084	0.1265	0.0086	0.3330	0.1039
1:4	SSFWAdaBoost	0.0793	0.0085	0.1090	0.0042	0.3212	0.0945
	FWAdaBoost	0.0672	0.0077	0.1275	0.0065	0.3194	0.0967
	Baseline	0.0796	0.0113	0.1201	0.0064	0.3308	0.1035

Table S36. Performance demonstration of SSFWAdaBoost-eClass0 and SSFWAdaBoost-ALMMo1 over 12 benchmark problems under splitting ratio 1:19 with $\frac{1}{3}$ of pseudo-labelling errors made by the initial base classifier $h_0(\boldsymbol{x})$ randomly corrected for training the remaining ensemble components

EFS	Algorithm	SE	LR	OR	PR	PW	MF
eClass0	SSFWAdaBoost$^{1/3}$	0.2128	0.3733	0.0964	0.1097	0.1412	0.1707
	SSFWAdaBoost	0.2466	0.4009	0.0795	0.1320	0.1747	0.2043
ALMMo1	SSFWAdaBoost$^{1/3}$	0.1254	0.3260	0.0868	0.0974	0.0559	0.0591
	SSFWAdaBoost	0.1801	0.4542	0.1050	0.1482	0.0839	0.0501
	SSFWAdaBoost	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
eClass0	SSFWAdaBoost$^{1/3}$	0.3179	0.1960	0.2678	0.1489	0.4553	0.2191
	SSFWAdaBoost	0.3994	0.0992	0.3200	0.1675	0.5691	0.2532
ALMMo1	SSFWAdaBoost$^{1/3}$	0.0496	0.0189	0.2369	0.0217	0.3062	0.1409
	SSFWAdaBoost	0.0761	0.0266	0.4854	0.0201	0.4468	0.1708

S6. Experimental Results under the Inductive Setting

Table S37. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:11:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.3167	0.2424	0.0354	0.0358	0.1330	0.1551
FWAdaBoost	0.3696	0.2748	0.0587	0.0526	0.1191	0.1745
AdaBoost.M1	0.3851	0.3151	0.0617	0.0538	0.1363	0.2038
AdaBoost.M2	0.4121	0.3229	0.0558	0.0554	0.1184	0.2254
SAMME	0.4047	0.3189	0.0726	0.0531	0.1367	0.2188
SAMME.R	0.4140	0.3147	0.0615	0.0618	0.1184	0.2055
RobAdaBoost	0.3377	0.2683	0.0495	0.0457	0.1121	0.1764
ASSEMBLE	0.2123	0.2369	0.0385	0.0394	0.1202	0.1617
SemiBoost	0.2732	0.2814	0.0389	0.0389	0.1166	0.2205
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0541	0.0271	0.2943	0.0439	0.3756	0.2906
FWAdaBoost	0.0599	0.0310	0.3024	0.0505	0.3927	0.3544
AdaBoost.M1	0.0573	0.0288	0.3053	0.0579	0.4276	0.3666
AdaBoost.M2	0.0583	0.0287	0.3449	0.0555	0.4457	0.4269
SAMME	0.0590	0.0288	0.3416	0.0770	0.4290	0.3934
SAMME.R	0.0981	0.0300	0.3281	0.0567	0.4149	0.3817
RobAdaBoost	0.0533	0.0268	0.2810	0.0483	0.3913	0.3065
ASSEMBLE	0.0648	0.0327	0.2917	0.0469	0.3659	0.1937
SemiBoost	0.1093	0.0977	0.2455	0.2693	0.4826	0.2485

Table S38. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:5:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2221	0.1601	0.0286	0.0235	0.1007	0.1114
FWAdaBoost	0.2738	0.1833	0.0402	0.0341	0.0988	0.1232
AdaBoost.M1	0.2975	0.2113	0.0421	0.0349	0.1137	0.1422
AdaBoost.M2	0.3383	0.2206	0.0386	0.0373	0.1004	0.1579
SAMME	0.3091	0.2113	0.0482	0.0375	0.1137	0.1435
SAMME.R	0.3019	0.2052	0.0410	0.0366	0.0979	0.1461
RobAdaBoost	0.2529	0.1771	0.0354	0.0302	0.0939	0.1236
ASSEMBLE	0.1527	0.1526	0.0309	0.0255	0.1014	0.1169
SemiBoost	0.1920	0.2119	0.0285	0.0216	0.0995	0.1435
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0479	0.0162	0.1760	0.0319	0.3220	0.2232
FWAdaBoost	0.0500	0.0186	0.2163	0.0368	0.3402	0.2709
AdaBoost.M1	0.0545	0.0203	0.2250	0.0418	0.3725	0.2911
AdaBoost.M2	0.0511	0.0190	0.2359	0.0379	0.3789	0.3327
SAMME	0.0550	0.0203	0.2352	0.0467	0.3719	0.3075
SAMME.R	0.0654	0.0189	0.2257	0.0372	0.3545	0.3106
RobAdaBoost	0.0490	0.0158	0.1995	0.0350	0.3320	0.2500
ASSEMBLE	0.0544	0.0180	0.2144	0.0333	0.3071	0.1550
SemiBoost	0.0850	0.0867	0.1812	0.2039	0.4439	0.1962

Table S39. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 3:9:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1850	0.1223	0.0250	0.0177	0.0922	0.0864
FWAdaBoost	0.2315	0.1386	0.0325	0.0243	0.0883	0.0940
AdaBoost.M1	0.2643	0.1653	0.0340	0.0244	0.1016	0.1162
AdaBoost.M2	0.3014	0.1696	0.0303	0.0250	0.0904	0.1104
SAMME	0.2785	0.1653	0.0423	0.0242	0.1016	0.1191
SAMME.R	0.2619	0.1569	0.0306	0.0244	0.0879	0.1053
RobAdaBoost	0.2056	0.1342	0.0291	0.0204	0.0860	0.0925
ASSEMBLE	0.1523	0.1125	0.0252	0.0176	0.0950	0.1030
SemiBoost	0.1668	0.1684	0.0243	0.0159	0.0901	0.1127
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0439	0.0139	0.1551	0.0236	0.2798	0.1908
FWAdaBoost	0.0449	0.0178	0.1780	0.0272	0.3044	0.2210
AdaBoost.M1	0.0510	0.0182	0.1860	0.0321	0.3364	0.2443
AdaBoost.M2	0.0481	0.0176	0.1881	0.0276	0.3698	0.2883
SAMME	0.0511	0.0197	0.2083	0.0417	0.3370	0.2632
SAMME.R	0.0577	0.0165	0.1813	0.0278	0.3205	0.2519
RobAdaBoost	0.0442	0.0156	0.1597	0.0253	0.2983	0.2048
ASSEMBLE	0.0496	0.0137	0.1617	0.0248	0.2648	0.1336
SemiBoost	0.0714	0.0752	0.1462	0.1541	0.4055	0.1680

Table S40. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFIS as base classifier over 12 benchmark problems under splitting ratio 1:2:2

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1698	0.1015	0.0206	0.0140	0.0854	0.0795
FWAdaBoost	0.1968	0.1146	0.0283	0.0199	0.0840	0.0901
AdaBoost.M1	0.2382	0.1337	0.0294	0.0191	0.0953	0.1018
AdaBoost.M2	0.2564	0.1388	0.0260	0.0196	0.0849	0.1029
SAMME	0.2463	0.1337	0.0322	0.0191	0.0953	0.1061
SAMME.R	0.2233	0.1284	0.0267	0.0199	0.0822	0.0980
RobAdaBoost	0.1876	0.1108	0.0238	0.0159	0.0798	0.0873
ASSEMBLE	0.1276	0.0936	0.0213	0.0142	0.0887	0.0910
SemiBoost	0.1501	0.1403	0.0229	0.0128	0.0818	0.1174
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0408	0.0125	0.1212	0.0199	0.2554	0.1612
FWAdaBoost	0.0439	0.0153	0.1499	0.0223	0.2759	0.1861
AdaBoost.M1	0.0477	0.0163	0.1659	0.0268	0.3093	0.2036
AdaBoost.M2	0.0443	0.0147	0.1639	0.0247	0.3270	0.2399
SAMME	0.0476	0.0158	0.1827	0.0315	0.3095	0.2144
SAMME.R	0.0487	0.0143	0.1527	0.0225	0.2965	0.2094
RobAdaBoost	0.0413	0.0137	0.1361	0.0207	0.2740	0.1744
ASSEMBLE	0.0475	0.0117	0.1386	0.0210	0.2435	0.1179
SemiBoost	0.0587	0.0628	0.1239	0.1220	0.3618	0.1417

Table S41. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:11:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2877	0.2184	0.0369	0.0304	0.1437	0.1329
FWAdaBoost	0.3244	0.2576	0.0589	0.0438	0.1312	0.1579
AdaBoost.M1	0.3265	0.3027	0.0650	0.0482	0.1408	0.1654
AdaBoost.M2	0.3909	0.3146	0.0572	0.0466	0.1477	0.2206
SAMME	0.3566	0.3288	0.0860	0.0578	0.1417	0.1906
SAMME.R	0.3390	0.2821	0.0564	0.0450	0.1307	0.1724
RobAdaBoost	0.3139	0.2499	0.0487	0.0394	0.1358	0.1604
ASSEMBLE	0.2245	0.2442	0.0340	0.0366	0.1312	0.1501
SemiBoost	0.2788	0.3030	0.0387	0.0402	0.1181	0.2324
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0543	0.0246	0.2826	0.0342	0.3621	0.2657
FWAdaBoost	0.0573	0.0292	0.3290	0.0521	0.4008	0.3014
AdaBoost.M1	0.0601	0.0306	0.3474	0.0586	0.4418	0.3147
AdaBoost.M2	0.0598	0.0291	0.4342	0.0572	0.4737	0.3662
SAMME	0.0658	0.0286	0.4041	0.0704	0.4482	0.3522
SAMME.R	0.1023	0.0270	0.3394	0.0515	0.4270	0.3442
RobAdaBoost	0.0538	0.0261	0.3185	0.0464	0.4175	0.2803
ASSEMBLE	0.0648	0.0327	0.2992	0.0511	0.3756	0.2071
SemiBoost	0.1453	0.0977	0.2670	0.3013	0.5027	0.2520

Table S42. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:5:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.2085	0.1435	0.0262	0.0195	0.1104	0.0995
FWAdaBoost	0.2425	0.1740	0.0400	0.0281	0.1019	0.1146
AdaBoost.M1	0.2568	0.2186	0.0440	0.0301	0.1117	0.1274
AdaBoost.M2	0.2979	0.2278	0.0375	0.0280	0.1150	0.1340
SAMME	0.2880	0.2189	0.0557	0.0386	0.1109	0.1489
SAMME.R	0.2793	0.1894	0.0368	0.0293	0.1013	0.1245
RobAdaBoost	0.2340	0.1670	0.0339	0.0245	0.1056	0.1164
ASSEMBLE	0.1902	0.1542	0.0257	0.0228	0.1056	0.1099
SemiBoost	0.2050	0.2424	0.0283	0.0234	0.1010	0.1685
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0498	0.0157	0.2039	0.0281	0.3035	0.2042
FWAdaBoost	0.0506	0.0177	0.2392	0.0370	0.3502	0.2287
AdaBoost.M1	0.0553	0.0198	0.2642	0.0426	0.3904	0.2677
AdaBoost.M2	0.0530	0.0182	0.2719	0.0393	0.4340	0.2929
SAMME	0.0586	0.0197	0.2947	0.0534	0.4002	0.2832
SAMME.R	0.0604	0.0177	0.2433	0.0359	0.3629	0.2539
RobAdaBoost	0.0488	0.0172	0.2267	0.0338	0.3550	0.2393
ASSEMBLE	0.0603	0.0198	0.2250	0.0378	0.3228	0.1869
SemiBoost	0.1049	0.0882	0.2058	0.2495	0.4785	0.2081

Table S43. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 3:9:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1663	0.1117	0.0218	0.0158	0.0987	0.0830
FWAdaBoost	0.2025	0.1356	0.0304	0.0207	0.0932	0.0892
AdaBoost.M1	0.2342	0.1714	0.0379	0.0248	0.1020	0.1029
AdaBoost.M2	0.2570	0.1719	0.0301	0.0216	0.1030	0.1025
SAMME	0.2494	0.1777	0.0437	0.0302	0.1026	0.1109
SAMME.R	0.2147	0.1427	0.0290	0.0209	0.0932	0.0958
RobAdaBoost	0.1937	0.1282	0.0275	0.0176	0.0952	0.0889
ASSEMBLE	0.1573	0.1190	0.0225	0.0166	0.0954	0.0955
SemiBoost	0.1831	0.2034	0.0253	0.0179	0.0914	0.1311
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0478	0.0132	0.1694	0.0211	0.2743	0.1685
FWAdaBoost	0.0447	0.0172	0.2027	0.0274	0.3177	0.1989
AdaBoost.M1	0.0522	0.0188	0.2214	0.0323	0.3648	0.2276
AdaBoost.M2	0.0502	0.0174	0.2168	0.0310	0.3980	0.2510
SAMME	0.0532	0.0186	0.2560	0.0437	0.3716	0.2479
SAMME.R	0.0561	0.0167	0.2119	0.0274	0.3397	0.2151
RobAdaBoost	0.0464	0.0151	0.1932	0.0244	0.3259	0.1936
ASSEMBLE	0.0528	0.0157	0.1917	0.0276	0.2939	0.1531
SemiBoost	0.0866	0.0801	0.1733	0.2129	0.4633	0.1789

Table S44. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with ALMMo0 as base classifier over 12 benchmark problems under splitting ratio 1:2:2

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1491	0.0934	0.0194	0.0117	0.0878	0.0759
FWAdaBoost	0.1779	0.1105	0.0269	0.0169	0.0837	0.0841
AdaBoost.M1	0.2069	0.1484	0.0300	0.0190	0.0895	0.0974
AdaBoost.M2	0.2363	0.1421	0.0251	0.0167	0.0910	0.0960
SAMME	0.2368	0.1604	0.0352	0.0207	0.0901	0.1212
SAMME.R	0.1946	0.1158	0.0248	0.0162	0.0847	0.0906
RobAdaBoost	0.1760	0.1054	0.0231	0.0139	0.0841	0.0849
ASSEMBLE	0.1462	0.0997	0.0200	0.0134	0.0910	0.0843
SemiBoost	0.1694	0.1795	0.0231	0.0136	0.0836	0.1196
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0430	0.0125	0.1440	0.0167	0.2521	0.1470
FWAdaBoost	0.0423	0.0143	0.1689	0.0220	0.2939	0.1739
AdaBoost.M1	0.0519	0.0176	0.1912	0.0286	0.3421	0.2009
AdaBoost.M2	0.0485	0.0160	0.1980	0.0257	0.3728	0.2183
SAMME	0.0520	0.0180	0.2290	0.0324	0.3568	0.2090
SAMME.R	0.0416	0.0141	0.1727	0.0216	0.3115	0.1847
RobAdaBoost	0.0426	0.0138	0.1664	0.0199	0.3017	0.1681
ASSEMBLE	0.0491	0.0126	0.1664	0.0218	0.2685	0.1302
SemiBoost	0.0705	0.0741	0.1576	0.1775	0.4430	0.1575

Table S45. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:11:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1531	0.2164	0.0382	0.0329	0.1040	0.0851
FWAdaBoost	0.1680	0.2588	0.0627	0.0454	0.1130	0.1204
AdaBoost.M1	0.1673	0.2441	0.0660	0.0425	0.1133	0.1209
AdaBoost.M2	0.1868	0.2510	0.0609	0.0420	0.1154	0.1321
SAMME	0.1983	0.2452	0.0894	0.0526	0.1127	0.1620
SAMME.R	0.6018	0.3868	0.2623	0.2496	0.1898	0.3549
RobAdaBoost	0.1552	0.2405	0.0579	0.0377	0.1105	0.0995
ASSEMBLE	0.1834	0.2229	0.0425	0.0356	0.1112	0.1043
SemiBoost	0.1984	0.2857	0.0369	0.0355	0.1099	0.1179
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0762	0.0286	0.2504	0.0684	0.3511	0.1371
FWAdaBoost	0.0818	0.0300	0.2998	0.1005	0.3530	0.1553
AdaBoost.M1	0.0813	0.0376	0.3195	0.1063	0.3499	0.1611
AdaBoost.M2	0.0795	0.0307	0.3846	0.1012	0.3902	0.1649
SAMME	0.0847	0.0389	0.3658	0.1294	0.3577	0.1862
SAMME.R	0.0990	0.0442	0.5750	0.5162	0.5098	0.4352
RobAdaBoost	0.0788	0.0290	0.3046	0.0868	0.3484	0.1416
ASSEMBLE	0.0856	0.0273	0.2912	0.0693	0.3650	0.1744
SemiBoost	0.1531	0.0925	0.2620	0.2275	0.4706	0.1836

Table S46. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:5:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.1038	0.1446	0.0315	0.0219	0.0844	0.0571
FWAdaBoost	0.1128	0.1779	0.0454	0.0275	0.0858	0.0771
AdaBoost.M1	0.1080	0.1611	0.0476	0.0285	0.0852	0.0792
AdaBoost.M2	0.1185	0.1738	0.0427	0.0265	0.0894	0.0700
SAMME	0.1258	0.1627	0.0600	0.0347	0.0852	0.1021
SAMME.R	0.5449	0.2795	0.2335	0.1255	0.2189	0.4245
RobAdaBoost	0.1019	0.1645	0.0435	0.0238	0.0847	0.0672
ASSEMBLE	0.1245	0.1402	0.0327	0.0253	0.0908	0.0689
SemiBoost	0.1251	0.2143	0.0306	0.0204	0.0911	0.0786
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0685	0.0147	0.1840	0.0466	0.2899	0.1001
FWAdaBoost	0.0685	0.0154	0.2171	0.0640	0.2927	0.1145
AdaBoost.M1	0.0677	0.0221	0.2386	0.0656	0.2873	0.1057
AdaBoost.M2	0.0733	0.0172	0.2421	0.0593	0.3121	0.1171
SAMME	0.0735	0.0212	0.2876	0.0790	0.2871	0.1166
SAMME.R	0.0977	0.2930	0.7074	0.4030	0.4143	0.6788
RobAdaBoost	0.0691	0.0157	0.2121	0.0540	0.2859	0.1029
ASSEMBLE	0.0723	0.0144	0.2192	0.0453	0.3062	0.1284
SemiBoost	0.1207	0.0808	0.2022	0.1576	0.4321	0.1364

Table S47. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 3:9:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.0837	0.1138	0.0274	0.0192	0.0718	0.0498
FWAdaBoost	0.0957	0.1412	0.0362	0.0212	0.0738	0.0611
AdaBoost.M1	0.0904	0.1218	0.0398	0.0185	0.0745	0.0658
AdaBoost.M2	0.0975	0.1321	0.0348	0.0193	0.0774	0.0615
SAMME	0.1055	0.1239	0.0464	0.0249	0.0745	0.0831
SAMME.R	0.4953	0.3063	0.2234	0.1322	0.2382	0.2410
RobAdaBoost	0.0871	0.1223	0.0367	0.0182	0.0738	0.0575
ASSEMBLE	0.1081	0.1067	0.0286	0.0200	0.0802	0.0644
SemiBoost	0.1068	0.1732	0.0284	0.0166	0.0825	0.0655
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0645	0.0093	0.1625	0.0390	0.2552	0.0858
FWAdaBoost	0.0660	0.0111	0.1769	0.0520	0.2554	0.0869
AdaBoost.M1	0.0633	0.0156	0.1959	0.0511	0.2475	0.0856
AdaBoost.M2	0.0694	0.0119	0.2017	0.0464	0.2650	0.0925
SAMME	0.0661	0.0152	0.2251	0.0642	0.2485	0.1043
SAMME.R	0.1007	0.2011	0.4440	0.3001	0.4506	0.5314
RobAdaBoost	0.0653	0.0107	0.1755	0.0425	0.2495	0.0828
ASSEMBLE	0.0698	0.0093	0.1736	0.0377	0.2645	0.1018
SemiBoost	0.1035	0.0717	0.1813	0.1212	0.3932	0.1016

Table S48. Performance comparison between SSFWAdaBoost and alternative boosting algorithms with SOFBIS as base classifier over 12 benchmark problems under splitting ratio 1:2:2

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost	0.0700	0.0938	0.0252	0.0158	0.0680	0.0461
FWAdaBoost	0.0765	0.1145	0.0305	0.0169	0.0687	0.0571
AdaBoost.M1	0.0740	0.1000	0.0319	0.0157	0.0676	0.0578
AdaBoost.M2	0.0784	0.1108	0.0290	0.0153	0.0707	0.0545
SAMME	0.0876	0.1023	0.0355	0.0169	0.0676	0.0769
SAMME.R	0.7016	0.2641	0.0447	0.2096	0.1518	0.4881
RobAdaBoost	0.0694	0.1016	0.0297	0.0147	0.0665	0.0499
ASSEMBLE	0.0845	0.0864	0.0251	0.0168	0.0778	0.0597
SemiBoost	0.0791	0.1439	0.0276	0.0131	0.0735	0.0569
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost	0.0583	0.0083	0.1341	0.0312	0.2327	0.0714
FWAdaBoost	0.0598	0.0097	0.1482	0.0420	0.2289	0.0753
AdaBoost.M1	0.0607	0.0120	0.1625	0.0381	0.2232	0.0743
AdaBoost.M2	0.0609	0.0092	0.1757	0.0364	0.2440	0.0779
SAMME	0.0614	0.0122	0.2050	0.0470	0.2232	0.0834
SAMME.R	0.0993	0.1187	0.2711	0.4235	0.3857	0.6930
RobAdaBoost	0.0596	0.0091	0.1507	0.0348	0.2244	0.0723
ASSEMBLE	0.0639	0.0078	0.1531	0.0288	0.2362	0.0889
SemiBoost	0.0865	0.0614	0.1664	0.0923	0.3521	0.0808

Table S49. Average ranks of classification performances of SSFWAdaBoost and alternative boosting algorithms over 12 benchmark problems

EFS	**Algorithm**	**Splitting Ratio**			
		1:19	**1:9**	**3:17**	**1:4**
SOFIS	SSFWAdaBoost	**2.4167**	**2.1667**	**2.3333**	**2.0833**
	FWAdaBoost	4.9167	4.2500	4.5833	4.8750
	AdaBoost.M1	6.1250	6.6250	7.0417	6.9583
	AdaBoost.M2	6.7083	7.2500	6.8333	6.6667
	SAMME	7.2083	7.5833	7.7500	7.5417
	SAMME.R	6.6250	6.1667	5.6250	5.7083
	RobAdaBoost	2.8333	3.0000	3.0000	2.9167
	ASSEMBLE	3.0833	2.7500	2.6667	2.7500
	SemiBoost	5.0833	5.2083	5.1667	5.5000
ALMMo0	SSFWAdaBoost	**2.1667**	**1.8333**	**1.7500**	**1.7500**
	FWAdaBoost	4.5417	4.2083	4.0833	4.3333
	AdaBoost.M1	6.4167	6.7917	7.0000	7.0000
	AdaBoost.M2	7.3333	7.0000	7.0000	6.8750
	SAMME	7.5833	7.6667	8.0000	8.2500
	SAMME.R	5.3333	5.2083	5.1667	4.5833
	RobAdaBoost	3.3333	3.4583	3.1667	3.4167
	ASSEMBLE	2.8750	3.1667	3.0000	3.0833
	SemiBoost	5.4167	5.6667	5.8333	5.7083
SOFBIS	SSFWAdaBoost	**1.3333**	**1.8750**	**1.9583**	**2.3333**
	FWAdaBoost	4.9167	4.7083	4.6250	4.8750
	AdaBoost.M1	5.0833	4.7083	4.1250	4.1667
	AdaBoost.M2	5.8333	5.4167	5.3333	4.9167
	SAMME	6.7500	6.2083	6.2917	6.2917
	SAMME.R	8.8333	8.8333	8.9167	9.0000
	RobAdaBoost	2.8333	2.9167	2.8750	2.5833
	ASSEMBLE	3.7500	4.0833	4.5417	4.7500
	SemiBoost	5.6667	6.2500	6.3333	6.0833

Table S50. *p*-values returned by Friedman tests for evaluating the statistical significance of SSFWAdaBoost over alternative boosting algorithms

EFS	**SOFIS**	**ALMMo0**	**SOFBIS**
p-value	0.0000	0.0000	0.0000

Table S51. *p*-values returned by pairwise Wilcoxon signed rank tests for evaluating the statistical significance of SSFWAdaBoost over alternative boosting algorithms

SSFWAdaBoost versus	**EFS**		
	SOFIS	**ALMMo0**	**SOFBIS**
FWAdaBoost	0.0000	0.0000	0.0000
AdaBoost.M1	0.0000	0.0000	0.0000
AdaBoost.M2	0.0000	0.0000	0.0000
SAMME	0.0000	0.0000	0.0000
SAMME.R	0.0000	0.0000	0.0000
RobAdaBoost	0.0000	0.0000	0.0000
ASSEMBLE	0.4571	0.0116	0.0000
SemiBoost	0.0018	0.0000	0.0000

Table S52. Performance comparison between SSFWAdaBoost-SOFBIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFIS and alternative semi-supervised ensemble classification models over 12 benchmark problems under splitting ratio 1:11:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.3167	0.2424	0.0354	0.0358	0.1330	0.1551
SSFWAdaBoost-ALMMo0	0.2877	0.2184	0.0369	0.0304	0.1437	0.1329
SSFWAdaBoost-SOFBIS	0.1531	0.2164	0.0382	0.0329	0.1040	0.0851
TriTrain-DT	0.1390	0.3838	0.2577	0.1616	0.0903	0.1759
TriTrain-kNN	0.2361	0.3219	0.0361	0.0480	0.1290	0.2034
ASSEMBLE-DT	0.1756	0.2737	0.0641	0.0458	0.0930	0.1522
ASSEMBLE-kNN	0.2108	0.2152	0.0387	0.0435	0.1232	0.1651
SemiBoost-DT	0.1516	0.3198	0.0789	0.0517	0.1033	0.2388
SemiBoost-kNN	0.2134	0.2892	0.0412	0.0369	0.1203	0.2406
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0541	0.0271	0.2943	0.0439	0.3756	0.2906
SSFWAdaBoost-ALMMo0	0.0543	0.0246	0.2826	0.0342	0.3621	0.2657
SSFWAdaBoost-SOFBIS	0.0762	0.0286	0.2504	0.0684	0.3511	0.1371
TriTrain-DT	0.0599	0.0275	0.5375	0.2056	0.4322	0.1390
TriTrain-kNN	0.0771	0.0417	0.3342	0.0850	0.4115	0.2372
ASSEMBLE-DT	0.0711	0.0225	0.3749	0.0945	0.3758	0.1554
ASSEMBLE-kNN	0.0769	0.0429	0.2579	0.0735	0.3558	0.2032
SemiBoost-DT	0.0923	0.0972	0.3036	0.2964	0.4617	0.1545
SemiBoost-kNN	0.1523	0.1019	0.2714	0.2885	0.4910	0.2022

Table S53. Performance comparison between SSFWAdaBoost-SOFBIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFIS and alternative semi-supervised ensemble classification models over 12 benchmark problems under splitting ratio 1:5:4

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.2221	0.1601	0.0286	0.0235	0.1007	0.1114
SSFWAdaBoost-ALMMo0	0.2085	0.1435	0.0262	0.0195	0.1104	0.0995
SSFWAdaBoost-SOFBIS	0.1038	0.1446	0.0315	0.0219	0.0844	0.0571
TriTrain-DT	0.1094	0.3177	0.2115	0.1227	0.0806	0.1149
TriTrain-kNN	0.1886	0.2052	0.0282	0.0279	0.1062	0.1484
ASSEMBLE-DT	0.1047	0.1766	0.0505	0.0320	0.0788	0.0751
ASSEMBLE-kNN	0.1518	0.1409	0.0258	0.0257	0.1069	0.1224
SemiBoost-DT	0.0903	0.2514	0.0651	0.0345	0.0876	0.1346
SemiBoost-kNN	0.1547	0.2287	0.0298	0.0205	0.1040	0.1737
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0479	0.0162	0.1760	0.0319	0.3220	0.2232
SSFWAdaBoost-ALMMo0	0.0498	0.0157	0.2039	0.0281	0.3035	0.2042
SSFWAdaBoost-SOFBIS	0.0685	0.0147	0.1840	0.0466	0.2899	0.1001
TriTrain-DT	0.0524	0.0152	0.4297	0.1783	0.3663	0.0969
TriTrain-kNN	0.0675	0.0189	0.2320	0.0524	0.3467	0.1763
ASSEMBLE-DT	0.0554	0.0103	0.3063	0.0753	0.2868	0.0976
ASSEMBLE-kNN	0.0691	0.0194	0.1925	0.0420	0.2925	0.1522
SemiBoost-DT	0.0660	0.0841	0.2353	0.2120	0.3947	0.0870
SemiBoost-kNN	0.1183	0.0949	0.2116	0.2389	0.4597	0.1571

Table S54. Performance comparison between SSFWAdaBoost-SOFBIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFIS and alternative semi-supervised ensemble classification models over 12 benchmark problems under splitting ratio 3:9:8

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.1850	0.1223	0.0250	0.0177	0.0922	0.0864
SSFWAdaBoost-ALMMo0	0.1663	0.1117	0.0218	0.0158	0.0987	0.0830
SSFWAdaBoost-SOFBIS	0.0837	0.1138	0.0274	0.0192	0.0718	0.0498
TriTrain-DT	0.0910	0.2839	0.1764	0.1045	0.0747	0.0929
TriTrain-kNN	0.1564	0.1604	0.0270	0.0193	0.1005	0.1210
ASSEMBLE-DT	0.0900	0.1433	0.0481	0.0258	0.0729	0.0638
ASSEMBLE-kNN	0.1364	0.1094	0.0225	0.0184	0.0971	0.1066
SemiBoost-DT	0.0740	0.2032	0.0650	0.0304	0.0775	0.1159
SemiBoost-kNN	0.1366	0.1868	0.0254	0.0167	0.0956	0.1367
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0439	0.0139	0.1551	0.0236	0.2798	0.1908
SSFWAdaBoost-ALMMo0	0.0478	0.0132	0.1694	0.0211	0.2743	0.1685
SSFWAdaBoost-SOFBIS	0.0645	0.0093	0.1625	0.0390	0.2552	0.0858
TriTrain-DT	0.0434	0.0114	0.4077	0.1462	0.3029	0.0844
TriTrain-kNN	0.0620	0.0132	0.2006	0.0379	0.3092	0.1574
ASSEMBLE-DT	0.0490	0.0073	0.2538	0.0553	0.2411	0.0921
ASSEMBLE-kNN	0.0619	0.0118	0.1771	0.0378	0.2413	0.1254
SemiBoost-DT	0.0541	0.0738	0.1997	0.1572	0.3408	0.0715
SemiBoost-kNN	0.1015	0.0836	0.1940	0.2087	0.4206	0.1306

Table S55. Performance comparison between SSFWAdaBoost-SOFBIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFIS and alternative semi-supervised ensemble classification models over 12 benchmark problems under splitting ratio 1:2:2

Algorithm	**SE**	**LR**	**OR**	**PR**	**PW**	**MF**
SSFWAdaBoost-SOFIS	0.1698	0.1015	0.0206	0.0140	0.0854	0.0795
SSFWAdaBoost-ALMMo0	0.1491	0.0934	0.0194	0.0117	0.0878	0.0759
SSFWAdaBoost-SOFBIS	0.0700	0.0938	0.0252	0.0158	0.0680	0.0461
TriTrain-DT	0.0795	0.2598	0.1638	0.0883	0.0745	0.0834
TriTrain-kNN	0.1293	0.1371	0.0258	0.0143	0.0924	0.1029
ASSEMBLE-DT	0.0744	0.1229	0.0446	0.0231	0.0670	0.0501
ASSEMBLE-kNN	0.1016	0.0911	0.0214	0.0149	0.0916	0.0964
SemiBoost-DT	0.0531	0.1710	0.0620	0.0267	0.0682	0.1152
SemiBoost-kNN	0.1124	0.1583	0.0237	0.0137	0.0877	0.1271
Algorithm	**PB**	**SB**	**SH**	**TE**	**GP**	**IS**
SSFWAdaBoost-SOFIS	0.0408	0.0125	0.1212	0.0199	0.2554	0.1612
SSFWAdaBoost-ALMMo0	0.0430	0.0125	0.1440	0.0167	0.2521	0.1470
SSFWAdaBoost-SOFBIS	0.0583	0.0083	0.1341	0.0312	0.2327	0.0714
TriTrain-DT	0.0390	0.0092	0.3711	0.1333	0.2891	0.0759
TriTrain-kNN	0.0561	0.0074	0.1871	0.0342	0.2726	0.1242
ASSEMBLE-DT	0.0454	0.0074	0.2356	0.0489	0.2108	0.0684
ASSEMBLE-kNN	0.0581	0.0074	0.1436	0.0255	0.2100	0.1046
SemiBoost-DT	0.0450	0.0558	0.1782	0.1172	0.2925	0.0507
SemiBoost-kNN	0.0884	0.0709	0.1758	0.1770	0.3787	0.1048

Table S56. Average ranks of classification performances of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0, SSFWAdaBoost-SOFBIS and alternative semi-supervised classification models over 12 benchmark problems

Algorithm	**Splitting Ratio**			
	1:11:8	**1:5:4**	**3:9:8**	**1:2:2**
SSFWAdaBoost-SOFIS	4.4167	4.4167	4.5000	4.3750
SSFWAdaBoost-ALMMo0	3.8333	4.0000	3.9583	4.0417
SSFWAdaBoost-SOFBIS	**2.6667**	**3.2500**	**3.4167**	**3.5833**
TriTrain-DT	5.5833	5.7500	5.5833	6.1667
TriTrain-kNN	6.3333	6.0833	6.5417	6.0000
ASSEMBLE-DT	4.5833	4.0000	4.1667	4.1667
ASSEMBLE-kNN	4.5000	4.5833	4.0833	4.0833
SemiBoost-DT	6.5000	6.0833	6.0000	5.8333
SemiBoost-kNN	6.5833	6.8333	6.7500	6.7500

Table S57. *p*-values returned by pairwise Wilcoxon signed rank tests for evaluating the statistical significance of SSFWAdaBoost-SOFIS, SSFWAdaBoost-ALMMo0 and SSFWAdaBoost-SOFBIS over alternative semi-supervised classification models

SSFWAdaBoost	**EFS**		
versus	**SOFIS**	**ALMMo0**	**SOFBIS**
TriTrain-DT	0.0049	0.0023	0.0000
TriTrain-kNN	0.0053	0.0006	0.0000
ASSEMBLE-DT	0.6518	0.9959	0.0002
ASSEMBLE-kNN	0.3198	0.5485	0.0007
SemiBoost-DT	0.0282	0.0194	0.0000
SemiBoost-kNN	0.0010	0.0002	0.0000