IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 June 2023, accepted 10 July 2023, date of publication 13 July 2023, date of current version 20 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294982

RESEARCH ARTICLE

STDPboost: A Self-Training Method Based on Density Peaks and Improved Adaboost for Semi-Supervised Classification

XU LIN[1] **AND JUNNAN LI**[2]
[1]School of Computer Engineering, Anhui Sanlian University, Hefei, Anhui 230601, China
[2]School of Artificial Intelligence and Big Data, Chongqing Industry Polytechnic College, Chongqing 401120, China

Corresponding author: Junnan Li (2022139@ctbu.edu.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62006029, and in part by the Natural Science Foundation of Chongqing under Grant CSTB2022NSCQMSX0258.

ABSTRACT The self-training methods have been praised by extensive research in semi-supervised classification. Mislabeling is the main challenge in self-training methods. Multiple variations of self-training methods are recently proposed against mislabeling from the following one of two aspects: a) using heuristic rules to find high-confidence unlabeled samples that can easily be predicted correctly in each iteration; b) enhancing prediction performance by employing ensemble classifiers composed of multiple weak classifiers. Yet, they still suffer from the following issues: a); most strategies for finding high-confidence unlabeled samples heavily rely on parameters; b) almost all employed ensemble classifiers originally designed for supervised classifiers and may not be suitable for semi-supervised classification due to the limited number and unrepresented distribution of the initial labeled data; c) few can overcome mislabeling from the above two aspects at the same time. To advance the state of the art, a new self-training method based on density peaks clustering and improved Adaboost is presented and named as STDPboost. In the iterative self-taught process, a new density peaks clustering-based strategy is proposed to find high-confidence unlabeled samples and a new ensemble classifier named Adaboost$_{\mathrm{SEMI}}$ and more suitable for semi-supervised classification is proposed to predict high-confidence unlabeled samples, which overcomes mislabeling and the mentioned shortcomings of existing self-training methods. Intensive experiments on benchmark data sets have proven that STDPboost outperforms 7 state-of-the-art self-training methods in average classification accuracy of KNN classifier and CART classifier with the percentages of the initial labeled data from 10% to 50% due to further alleviating mislabeling.

INDEX TERMS Semi-supervised learning, semi-supervised classification, self-training methods, oversampling techniques, Adaboost.

I. INTRODUCTION

Supervised classification (SC) [1] is an active research field and has featured extensive practical applications, such as intelligent medical detection [2], target tracking [3], recommended system [4], and cross-language translation [5]. SC can use a supervised classification model learned from apriori knowledge (i.e., labeled samples) to predict unseen samples. It is widely known that the performance of supervised classification models in SC heavily depends on the breadth and depth of prior knowledge (i.e., the quality and quantity of labeled samples). Traditional SC needs manual annotation of data to obtain sufficient prior knowledge, which is a time-consuming and expensive job in the big data era. Semi-supervised classification (SSC) [6] has recently emerged to overcome the above bottleneck of data annotation in SC by exploiting both labeled and unlabeled data.

After decades of development, various semi-supervised classification paradigms are proposed by scholars, and they can be generally divided into algorithm-level SSC

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan.

methods [7], [8], [9] and data-level SSC methods [10]. Algorithm-level SSC methods modify traditional supervised classification models to make them suitable for SSC. Semi-supervised support vector machine (S3VM) classifiers [7], semi-supervised optimization path forest (OPFSEMI) [8] and semi-supervised k nearest neighbor classifiers [9] are representative instances of algorithm-level SSC methods. Data-level SSC methods improve the distribution and number of the initial labeled data by iterative annotating available unlabeled data. Self-training methods [10] are typical data-level SSC methods. Compared to other SSC methods, self-training methods are easier to understand and more simple but more effective. Hence, self-training methods are easier to be implemented in practical fields [11], [12], [13], [14], [15].

As shown in Fig. 1, Self-training methods are based on an iterative self-taught (or self-labeled) idea. In detail, self-training methods use classification models C learned from the set of labeled data L to iteratively predict high-confidence unlabeled samples X_{HCS} selected from the set of unlabeled data U. The predicted samples with pseudo labels L_{new} are added to the set of labeled data, and then classification models C are relearned from the extended set of labeled data L. The above self-taught process is repeated until all unlabeled samples are predicted. After that, self-training methods output a given classifier C trained on the improved set of labeled data L.

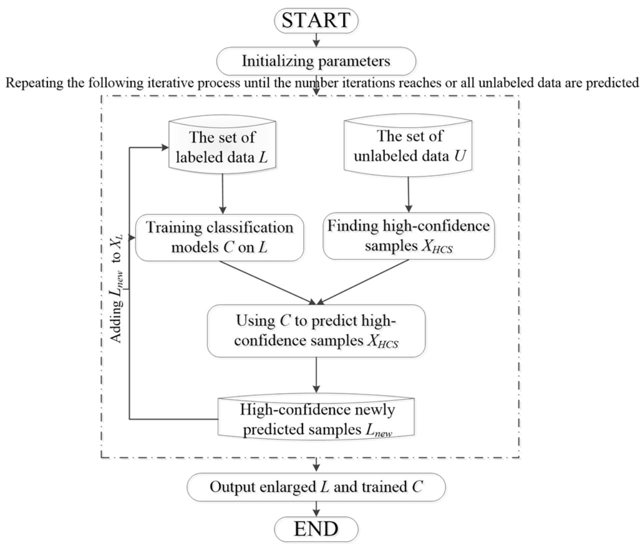
START
Initializing parameters
Repeating the following iterative process until the number iterations reaches or all unlabeled data are predicted
The set of labeled data L
The set of unlabeled data U
Training classification models C on L
Finding high-confidence samples X_{HCS}
Using C to predict high-confidence samples X_{HCS}
High-confidence newly predicted samples L_{new}
Adding L_{new} to X_L
Output enlarged L and trained C
END

FIGURE 1. A flowchart of the standard self-training method.

Evidently, self-training methods expect their prediction (i.e. the above pseudo labels) to be correct. However, self-training methods inevitably mispredict unlabeled data because the initial labeled data are usually rare, their distribution is usually unrepresented and the initial labeled data may contain noise or outliers. If mispredicted samples with pseudo-label noise are added to the set of labeled data, the performance of self-training methods will degrade, which leads to more mislabeling [16]. Hence, the mislabeling that refers to the mispredicting unlabeled samples is one of the main challenges in self-training methods [17]. To overcome mislabeling, multiple variations of self-training methods [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31] are recently developed and can be classified into the following two categories:

The first category (self-training method based on heuristic rules for finding high-confidence unlabeled samples) employs heuristic rules, such as the maximum posterior probability of classifiers, nearest-neighbor rules, clustering algorithms, and graph models, to find high-confidence unlabeled samples that can easily be predicted correctly in each iteration. This indirectly improves prediction accuracy, thus alleviating mislabeling. Typical instances are SETRED [18], SNNRCE [19], MLSTE [20], Help-Training [21], STS-FCM [22], STDP [23], STDPCEWS [24], STOPF [25] and NaNG-ST [26].

The second category (self-training methods based on ensemble classifiers) employs ensemble classifiers composed of multiple weak classifiers to enhance prediction for unlabeled samples in each iteration, which tends to directly alleviate mislabeling. Typical instances are Co-Bagging [27], Co-Adaboost [27], Co-Forest [28], Tri-training [29], Multi-Train [30], and BoostSTIG [31].

Despite their effectiveness, the above two categories of self-training methods still suffer from the following shortcomings:

(a) Firstly, most strategies for finding high-confidence unlabeled samples heavily rely on parameters in the first category of self-training variations, leading to unstable performance and difficulty in application.

(b) Secondly, almost all employed ensemble classifiers in the second category of self-training variations were originally designed for SC and may not be suitable for semi-supervised classification due to the limited number and unrepresented distribution of the initial labeled data, which may lead to the misprediction for unlabeled samples.

(c) Additionally, few use both strategies of finding high-confidence unlabeled samples and using ensemble classifiers to overcome mislabeling. Actually, the above two strategies are easy to combine and can work together to further alleviate mislabeling in self-training methods.

A. OBJECTIVES AND CONTRIBUTIONS

The main objective of this paper is to propose a new self-training method (named STDBboost) based on density peaks clustering and improved Adaboost. STDBboost can further overcome mislabeling by designing a new strategy for finding high-confidence unlabeled samples and a new ensemble classifier, which overcomes the mentioned above issues in the existing self-training methods.

The second objective of this paper is to propose a new ensemble classifier named Adaboost$_{\text{SEMI}}$. Ensemble classifiers in the second category of self-training variations are not fully compatible with SSC since they can not utilize a large number of unlabeled samples and only utilize a small number of labeled samples. Adaboost$_{\mathrm{SEMI}}$ combining the idea of oversampling techniques can generate synthetic labeled samples to improve the labeled set and are more suitable for SSC.

The third objective of this paper is to propose a new parameter-free strategy (named DPCStr) based on DPC (Density Peaks Clustering) [32] to find high-confidence unlabeled samples. Strategies for finding high-confidence unlabeled samples in existing self-training methods heavily rely on parameters. DPCStr helps STDBboost find high-confidence unlabeled samples without parameters.

The main contributions are highlighted as follows:

(a) A new self-training method named STDPboost is proposed against mislabeling and the mentioned shortcomings of existing self-training variations [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31].

(b) A new DPC-based strategy named DPCStr is proposed to find high-confidence unlabeled samples in each iteration of STDPboost.

(c) A new ensemble classifier named Adaboost$_{\mathrm{SEMI}}$ is proposed to predict unlabeled samples in each iteration of STDPboost. Although Adaboost$_{\mathrm{SEMI}}$ is slightly similar to SMOTEBoost [33] in some aspects, SMOTEBoost is mainly used for imbalanced classification and Adaboost$_{\mathrm{SEMI}}$ is mainly used for SSC. Besides, SMOTEBoost aims to generate synthetic minority class samples, while Adaboost$_{\mathrm{SEMI}}$ aims to synthetic labeled samples for all classes.

(d) Empirical results and conclusions with 7 state-of-the-art self-training methods and 2 popular classifiers on extensive UCI and Kaggle benchmark data sets are reported.

The rest of the paper is organized as follows. Related work is reviewed in Section II. Preliminaries are introduced in Section III. STDPboost is introduced in Section IV. The results of comparison experiments are reported in Section V. Section VI makes conclusions.

II. RELATED WORK

The proposed STDPboost intends to overcome mislabeling and the mentioned issues in two categories of self-training methods [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31] for overcoming mislabeling. Hence, in this section, only recent self-training methods, related solutions [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31] for overcoming mislabeling and comparative self-training methods are reviewed.

The earliest self-training method was first proposed by Yarowsky [34] in 1995 and applied to intelligent word sense disambiguation with a self-taught idea. The self-taught idea refers to using samples predicted by itself to teach itself. Although multiple variations of self-training methods are developed recently, some research [16], [17] has found that self-training methods inevitably mispredict unlabeled data due to the limited number and unrepresented distribution of initial labeled data. Once pseudo-label noise caused by misprediction is added to the set of labeled data, self-training methods are highly likely to mispredict more samples, leading to training an ineffective classifier. Solutions for overcoming mislabeling of self-training methods can be generally divided into self-training methods [18], [19], [20], [21], [22], [23], [24], [25], [26] based on heuristic rules for finding high-confidence unlabeled samples and self-training methods [27], [28], [29], [30], [31] based on ensemble classifiers.

A. SELF-TRAINING METHOD BASED ON FINDING HIGH-CONFIDENCE UNLABELED SAMPLES

SETRED (SElf-TRaining with EDited) [18], SNNRCE (Self-training Nearest Neighbor Rule using Cut Edges) [19], MLSTE (Multi-Label Self-Training with Editing) [20], Help-Training [21], STSFCM (Self-Training with Semi-supervised Fuzzy C-Means) [22], STDP (Self-Training with Density Peaks) [23], STDPCEWS (Self-Training with Density Peaks and Cut Edge Weight Statistic) [24], STOPF (Self-Training with Optimum-Path forest) [25] and NaNG-ST (Natural Neighborhood Graph-based Self-Training) [26] belong to the first category solution (self-training methods based on heuristic rules for finding high-confidence unlabeled samples). Some heuristic rules (e.g. the maximum posterior probability of classifiers, nearest-neighbor rules, clustering algorithms and graph models) are used to find high-confidence unlabeled samples that can easily be predicted correctly in each iteration of the first categories solution, which indirectly improves prediction accuracy and alleviates mislabeling.

SETRED and SNNRCE employ nearest-neighbor rules to find high-confidence unlabeled samples in each iteration, where high-confidence unlabeled samples are close to their k-nearest neighbors searched on one class, and far away from their k-nearest neighbors searched on other classes. MLSTE calculates a centroid of each class, and regards unlabeled samples close to one centroid as high-confidence unlabeled samples in each iteration. Help-Training uses the maximum posterior probability of Naive Bayes [35] to find high-confidence unlabeled samples in each iteration, where high-confidence unlabeled samples have a relatively high posterior probability. STSFCM, STDP and STDPCEWS employ clustering algorithms to find high-confidence unlabeled samples in each iteration. STSFCM executes SSFCM (Semi-Supervised Fuzzy C-means clustering) [36] in each iteration, where high-confidence unlabeled samples are close to clustering centers and have a relatively high fuzzy membership. STDP and STDPCEWS execute DPC (Density Peaks Clustering) [32] before the iteration of the self-taught process begins. Then, they use the assigning strategy of non-central samples to find high-confidence unlabeled samples in each iteration. Additionally, STOPF and NaNG-ST use heuristic graph models to find high-confidence unlabeled samples in each iteration. STOPF constructs an optimization path forest [8] on labeled and unlabeled training data. NaNG-ST constructs a natural neighborhood graph [37] on labeled and unlabeled training data. Then, they regard unlabeled samples connected with a labeled sample as high-confidence unlabeled samples. In each iteration of the self-taught process, the mentioned above self-training methods first finds high-confidence unlabeled samples, and then use high-confidence samples with pseudo labels predicted by a given single classifier to update the labeled training set iteratively.

In summary, the above self-training methods [18], [19], [20], [21], [22], [23], [24], [25], [26] based on heuristic rules for finding high-confidence unlabeled samples can alleviate mislabeling by finding easy-to-predict high-confidence unlabeled samples. Yet, most strategies (e.g. [18], [19], [20], [21], [22], [23], [24]) for finding high-confidence unlabeled samples heavily rely on parameters, leading to lower robustness. For instance, these strategies in SETRED, SNNRCE, MLSTE, STSFCM, STDP and STDPCEWS heavily rely on the parameter k of neighbors, the parameter d_c of DPC or/ and the confidence thresholds of finding top-n high-confidence unlabeled samples.

B. SELF-TRAINING METHOD BASED ON ENSEMBLE CLASSIFIERS

Co-Bagging [27], Co-Adaboost [27], Co-Forest [28], Tri-training [29], Multi-Train [30] and BoostSTIG [31] belong to the second category solution (self-training methods based on ensemble classifiers). Some ensemble classifiers composed of multiple weak classifiers are employed in them, aiming to directly alleviate mislabeling by enhancing prediction for unlabeled samples in each iteration. Co-Bagging or Co-Adaboost employs Bagging [38] or Adaboost [38] to construct a homogenous ensemble classifier in each iteration, and then Co-Bagging or Co-Adaboost uses the constructed ensemble classifier to predict unlabeled samples selected randomly from the unlabeled set. Co-Forest employs the random forest classifier [39] composed of multiple decision tree classifiers to predict unlabeled samples randomly selected from the unlabeled set in each iteration. The classification error rate of the sample is used to resample the training set to generate three different classifiers in Tri-training. In the iteration of Tri-training, an unlabeled example is labeled for a classifier if the other two classifiers agree on the labeling. Multi-train generates some homogenous or heterogeneous classifiers that use different classification models and/or different features. During the self-taught process, each classifier is refined using unlabeled data with the strategy of the majority vote. BoostSTIG first generates synthetic labeled samples to improve the set of initial labeled data. Then, it uses AdaBoost, MadaBoost [31], MultiBoost [31] or ReweightBoost [31] to train a homogenous ensemble classifier in each iteration, intending to improve the prediction for unlabeled samples.

In summary, the second category self-training methods [27], [28], [29], [30], [31] can alleviate mislabeling by using ensemble classifiers to improve prediction accuracy. Yet, almost all ensemble classifiers (e.g. [27], [28], [29], [30]) are not suitable for SSC because they only utilize a small number of labeled samples with an unrepresented distribution, easily leading to misprediction for unlabeled samples.

C. SUMMARIES AND DIFFERENCES BETWEEN STDPBOOST AND EXISTING METHODS

Generally, existing self-training variations use strategies for finding high-confidence unlabeled samples or ensemble classifiers to alleviate mislabeling. Nevertheless, as analyzed in Section I, they still suffer from the following issues: a) most strategies for finding high-confidence unlabeled samples heavily rely on parameters; b) almost all employed ensemble classifiers are not suitable for SSC due to the limited number and distribution of initial labeled data; c) few can overcome mislabeling by using both of strategy for finding high-confidence unlabeled samples and ensemble classifiers.

To this end, a new self-training method (STDPboost) based on density peaks clustering and improved Adaboost is presented in this paper. The main differences between STDPboost and each of the existing self-training methods are highlighted as follows:

(a) Compared to the first category of self-training variations [18], [19], [20], [21], [22], [23], [24], [25], [26], a new strategy (DPCStr) based on DPC is proposed in STDPboost and can find high-confidence unlabeled samples without parameters.

(b) Compared to the second category of self-training variations [27], [28], [29], [30], [31], a new ensemble classifier (Adaboost$_{\mathrm{SEMI}}$) is proposed in STDPboost and is more suitable for SSC due to generating synthetic labeled samples for all classes to improve the labeled set.

(c) Compared to the mentioned two categories of self-training variations [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], STDPboost can further overcome mislabeling by using both of a new strategy (DPCStr) for finding high-confidence unlabeled samples and a new ensemble classifier (Adaboost$_{\mathrm{SEMI}}$) in the self-training process.

III. PRELIMINARIES

This section describes the main notations and preliminaries, such as SMOTE [40] and Adaboost [34], which provides a theoretical basis for STDPboost.

A. NOTATIONS

The main notations of this paper are described as follows:

$X_{SSL} = \{x_1, x_2, \ldots, x_{n_L}\ x_{n_L+1}, \ldots, x_n\}$ is the data set with n labeled and unlabeled samples. $L = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_L}, y_{n_L},)\}$ and $U = \{x_{n_L+1}, x_{n_L+2}, \ldots, x_n\}$. y_i $(i = 1, \ldots, n)$ is the class label of x_i, where $y_i \in \omega = \{\omega_1, \ldots, \omega_K\}$. $X_{SSL} = L \cup U$, where L is the set of labeled data and U is the set of unlabeled data. $x_i = \{x_{i,1}, \ldots, x_{i,n_{attr}}\}$ is the ith sample with n_{attr} attributes.

- $NN_k(x_{base}, X_{input})$ is the set of k nearest neighbors of sample x_{base} and searched on X_{input}.
- x_{new} is a newly generated synthetic sample.
- X_{syn} is the set of synthetic minority class samples.
- f_m $(m = 1, 2, \ldots, M)$ is the weak classifier in the ensemble classifier F.

Algorithm 1 Synthetic Minority Oversampling Technique (*SMOTE*)

```
Input: $X_{input}$ (The set of input data), $N$ (The number of synthetic
samples for each minority class sample), $k$ (The parameter of                Time complexity
searching for $k$ nearest neighbors)
Output: $X_{syn}$ (The set of synthetic samples)
 1:  $X_{syn} = \emptyset$;                                                    $O(1)$
 2:  for $x_i \in X_{input}$                                                   $O(n_{input})$
 3:      $x_{base} = x_i$;                                                     $O(n_{input})$
 4:      $TempN = N$;                                                          $O(n_{input})$
 5:      while $TempN > 0$                                                     $O(N \times n_{input})$
 6:          $x_r$ is one of $NN_k(x_{base}, X)$ and is randomly selected;     $O(N \times n_{input} \times log(n_{input}))$
 7:          for $j = 1$ to $n_{attr}$                                         $O(N \times n_{input})$
 8:              $dif = x_{r,j} - x_{base,j}$;                                 $O(N \times n_{input})$
 9:              $x_{new,j} = x_{base,j} + rand(0, 1) \times dif$;             $O(N \times n_{input})$
10:          end for                                                           $O(N \times n_{input})$
11:          $X_{syn} = X_{syn} \cup \{x_{new}\}$;                             $O(N \times n_{input})$
12:          $TempN = TempN - 1$;                                              $O(N \times n_{input})$
13:      end while                                                             —
14:  end for                                                                   —
15:  return $X_{syn}$;                                                         O(1)
```

- a_m $(m = 1, 2, \ldots, M)$ is the weight of each weak classifier f_m.
- $W = \{w_1, \ldots\}$ is the sample distribution.
- $F(x_i, \omega_j)$ returns the prediction of the ensemble classifier F for sample x_i in class ω_j.
- $H(x_i)$ returns the class label of sample x_i by using the ensemble classifier F.
- $p(x_i)$ is the local density of sample x_i.
- $\delta(x_i)$ is the offset distance $\delta(x_i)$ of sample x_i.
- $next(x_i) = x_j$ denotes that sample x_i points to x_j in the DPC-based strategy (DPCStr).
- $X_{HCS} = \{x_j, \ldots\}$ is the set of high-confidence unlabeled samples.

B. SMOTE

SMOTE (Synthetic Minority Oversampling TEchnique) is proposed by Li et al. [40]. It aims to improve the class distribution by generating minority class synthetic samples, in which minority class synthetic samples are generated by the random interpolation between each minority class sample and one of its k nearest neighbors. The interpolation can be indicated by formula (1):

$$x_{new,j} = x_{base,j} + rand(0, 1) \times (x_{base,j} - x_{r,j}) \tag{1}$$

In formula (1), x_{new} is a newly generated synthetic sample. x_{base} is a base sample selected from the minority class. x_r is one of k nearest neighbors of x_{base}. The set of k nearest neighbors of x_{base} and searched on X_{input} is denoted as $NN_k(x_{base}, X_{input})$. $x_{new,j}$, $x_{base,j}$ or $x_{r,j}$ is the jth $(j = 1, \ldots, n_{atrr})$ attribute of x_{new}, x_{base} or x_r, respectively. The function $rand(0, 1)$ returns a random number between 0 and 1. The pseudo-code of SMOTE is described in Algorithm 1 which needs to set two parameters (i.e., N and k).

Algorithm 1 takes minority class samples as input data. At Lines 2-3, each minority class sample is regarded as a base sample. Lines 4-14 show the random interpolation process of SMOTE, where the class label of x_{new} is the same as that of x_{base}. Finally, Algorithm 1 returns the set of synthetic minority class samples X_{syn}. The time complexity of Algorithm 1 is $O(N \times n_{input} \times log(n_{input}))$, where N is the number of synthetic samples for each sample in X_{input} and n_{input} is the sample number in X_{input}. For more details on SMOTE, please refer to the work [40].

C. ADABOOST

The Adaboost [31] is one of the most successful methodologies for constructing an ensemble classifier denoted as F. The ensemble classifier usually has a better classification accuracy because it combines the performance of multiple weak classifiers f_m $(m = 1, 2, \ldots, M)$. M is the number of weak classifiers. As a general rule, Adaboost continuously modifies the sample distribution $\boldsymbol{W} = \{w_1, \ldots\}$, and then uses the sample distribution to train each weak classifier f_m. Finally, an ensemble classifier F can be constructed by formula (2).

$$F(x_i, \omega_j) = \sum_{m=1}^{M} (a_m \times f_m(x_i, \omega_j)) \tag{2}$$

$F(\boldsymbol{x}_i, \omega_j)$ returns the prediction of the ensemble classifier F for sample $\boldsymbol{x}_i$ in class ω_j. It combines the prediction of multiple weak classifiers linearly. The variable a_m $(m = 1, 2, \ldots, M)$ is the weight of each weak classifier f_m and is calculated by the classification error rate ε of f_m. The variable

Algorithm 2 *Adaboost*

```
Input: $X_{input}$ (The set of input data with $n_{input}$ sample), $M$ (The number of weak classifiers in Adaboost)    Time complexity
Output: $F$ (The trained ensemble classifier)
1:  $\forall x_i \in X, w_i = 1/n_{input}$;                                                          $O(n_{input})$
2:  for $m = 1$ to $M$                                                                             $O(M)$
3:      Training a classifier $f_m$ learned from $X'$ and $X'$ is obtained by using $W$ to resample $X_{input}$;   $O(M \times f)$
4:      Calculating the error rate $\varepsilon$ of $f_m$ by using formula (5);                    $O(M \times f)$
5:      if $\varepsilon > 0.5$                                                                     $O(M)$
6:          $a_m = 0$;                                                                             $O(M)$
7:          Reset $W$: $\forall x_i \in X_{input}, w_i = 1/n_{input}$;                             $O(M \times n_{input})$
8:      else if $\varepsilon == 0$                                                                 $O(M)$
9:          $a_m = 10$;                                                                            —
10:         Reset $W$: $\forall x_i \in X_{input}, w_i = 1/n_{input}$;                             —
11:     else                                                                                       —
12:         $a_m = (1/2) \times ln((1-\varepsilon)/\varepsilon)$;                                  $O(M)$
13:         for each $x_i \in X_{input}$                                                           $O(M \times n_{input})$
14:             if $f_m(x_i) = y_i$                                                                $O(M \times n_{input})$
15:                 $w_i = w_i/(2 \times (1-\varepsilon))$;                                        $O(M \times n_{input})$
16:             else                                                                               —
17:                 $w_i = w_i/2 \times \varepsilon$;                                              $O(M \times n_{input})$
18:             end if                                                                             —
19:         end for                                                                                —
20:     Normalizing $W$;                                                                           $O(M \times n_{input})$
21: end for                                                                                        —
22: return $F$ by using formula (2);                                                               $O(M)$
```

a_m $(m = 1, 2, \ldots, M)$ is calculated by formula (3).

$$a_m = \begin{cases} a_m = 0 & \text{if } \varepsilon > 0.5 \\ a_m = 10 & \text{if } \varepsilon == 0 \\ a_m = \frac{1}{2} \times ln(\frac{1-\varepsilon}{\varepsilon}) & \text{otherwise} \end{cases} \tag{3}$$

$H(\boldsymbol{x}_i)$ returns the class label of sample $\boldsymbol{x}_i$ by using the ensemble classifier F and can be calculated by formula (4).

$$H(\boldsymbol{x}_i) = \underset{\omega_j \in \boldsymbol{\omega}}{arg\,max}\,(F(x_i, \omega_j)) \tag{4}$$

The classification error rate ε of f_m can be calculated by formula (5).

$$\varepsilon = \frac{1}{n}\sum_{i=1}^{n} w_i \times I(f_m(x_i) \neq y_i) \tag{5}$$

If $I(f_m(x_i) \neq y_i)$, the function $I()$ returns 1, otherwise 0. Based on the general principle, the pseudo-code of Adaboost is described in Algorithm 2 which needs to set a parameter M.

At Line 1, the sample distribution w_i for each sample x_i in X_{input} is initialized. At Lines 2-21, the weight a_m of each weak classifier f_m is calculated, and then sample distribution w_i for each sample x_i in X_{input} is updated according to the error rate ε. Finally, the ensemble classifier F is obtained at Line 22. The time complexity of Algorithm 2 is O($M \times f$), where M is the number of weak classifiers and $O(f)$ is the time complexity of the adopted weak classifier. For more details on Adaboost, please refer to the work [31], [33], [38].

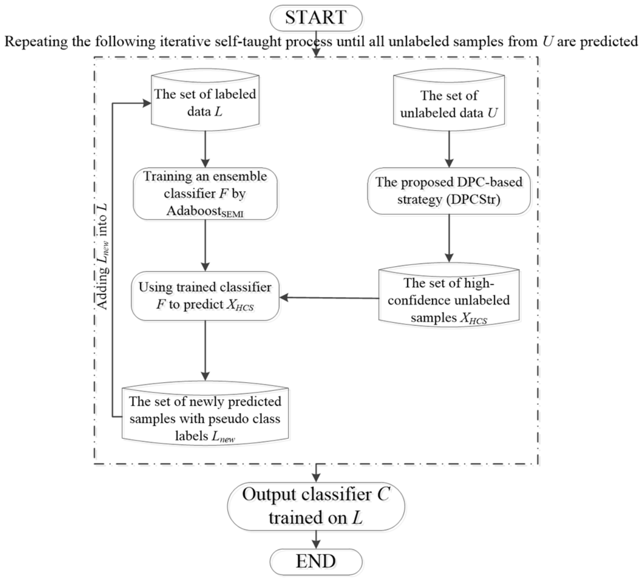
START
Repeating the following iterative self-taught process until all unlabeled samples from U are predicted
The set of labeled data L
The set of unlabeled data U
Training an ensemble classifier F by Adaboost$_{SEMI}$
The proposed DPC-based strategy (DPCStr)
Adding L_{new} into L
Using trained classifier F to predict X_{HCS}
The set of high-confidence unlabeled samples X_{HCS}
The set of newly predicted samples with pseudo class labels L_{new}
Output classifier C trained on L
END

FIGURE 2. A general flowchart of the proposed STDPboost.

IV. PROPOSED ALGORITHM

In this section, STDPboost is described at length. A general flowchart for STDPboost is shown in Fig. 2. As visually illustrated in Fig. 2, STDPboost is an iterative algorithm and includes the following iterative self-taught process: a) First, An ensemble classifier F is trained on the set of labeled data L by using the proposed Adaboost$_{\text{SEMI}}$; b) Second, the proposed DPC-based strategy (DPCStr) is used to find high-confidence unlabeled samples X_{HCS} from the set of unlabeled data U; c) Third, the trained ensemble classifier F is used to predict high-confidence unlabeled samples X_{HCS}; d) Four, predicted high-confidence samples with pseudo labels L_{new} are added to L. The above iterative self-taught process repeats until all unlabeled samples from U are predicted. After that, STDPboost outputs a given classifier C trained on the improved L.

In the following, Section IV-A introduces the proposed DPC-based strategy for finding high-confidence unlabeled samples. Section IV-B introduces the proposed Adaboost$_{\text{SEMI}}$. Section IV-C introduces the pseudo-code of the proposed STDPboost.

A. DPC-BASED STRATEGY FOR FINDING HIGH-CONFIDENCE UNLABELED SAMPLES

In STDPboost, a new parameter-free DPC-based strategy (DPCStr) is proposed to find high-confidence unlabeled samples that can easily be predicted correctly in each iteration.

The DPC can find data characteristics by calculating the local density and the offset distance of each sample. Then, it takes the sample with both a high local density and a high offset distance as the cluster center, where non-central samples are assigned to its nearest neighbor with a higher local density. Inspired by DPC, the proposed DPCStr employs the ideas of the local density, the offset distance and the assigning strategy of non-central samples to find high-confidence unlabeled samples.

First, DPCStr calculates the local density $p(x_i)$ of each sample x_i in X_{SSL} by formula (6).

$$p(x_i) = \sum_{x_j \in X_{SSL}} sign(dist(x_i, x_j) - d_c) \tag{6}$$

The function $dist()$ returns the Euclidean distance between two samples. The sign function $sign()$ is formulated by formula (7).

$$sign(t) = \begin{cases} 1 & t < 0 \\ 0 & t \geq 0 \end{cases} \tag{7}$$

In formula (6), the cutoff distance d_c is calculated by formula (8).

$$d_c = \max_{i=1}^{n} (\min_{j=1}^{n} (dist(x_i, x_j))) \tag{8}$$

Next, DPCStr calculates the offset distance $\delta(x_i)$ of each sample x_i in X_{SSL} by formula (9).

$$\delta(x_i) = \begin{cases} \min_{j=1}^{n} (dist(x_i, x_j)) & \forall x_j, p(x_i) < p(x_j) \\ \max_{j=1}^{n} (dist(x_i, x_j)) & \forall x_j, p(x_i) \geq p(x_j) \end{cases} \tag{9}$$

After calculating the local density and offset distance of each sample, DPCStr makes each sample point to its nearest neighbor with a higher local density, as shown in Fig. 3.

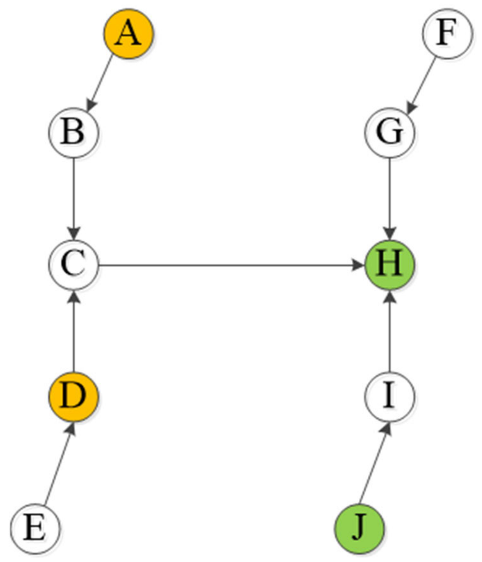

FIGURE 3. Using toy data to illustrate the proposed DPCStr.

Fig. 3 contains labeled samples of two classes with yellow or green circles and unlabeled samples with white hollow circles, where each sample points to its nearest sample with a higher local density. In DPCStr, if sample x_i points to sample x_j, we denote it as $next(x_i) = x_j$. For instance in Fig. 2, $next$(A) = B, $next$(B) = C, $next$(C) = H, $next$(D) = C and $next$(E) = D. Based on the above theory, the high-confidence unlabeled samples can be found by using Definition 1.

Definition 1 (High-Confidence Unlabeled Samples): Let $X_{HCS} = \{x_j, \ldots\}$ be the set of high-confidence unlabeled samples. If sample x_i belongs to X_{HCS}, a labeled sample points to sample x_i or sample x_i points to a labeled sample in each iteration of STDPboost. The set of high-confidence unlabeled samples can be calculated by formula (10).

$$X_{HCS} = \{x_i | (next(x_j) == x_i \text{ or } next(x_i) == x_j) \text{ and } x_j \in L\} \tag{10}$$

According to Definition 1, samples B, C, E, G and I belong to X_{HCS} because $next$(A) = B, $next$(D) = C, $next$(E) = D, $next$(J) = I and $next$(G) = H. According to the analysis in Fig. 3 and Definition 1, the high-confidence unlabeled samples are some unlabeled samples close to labeled samples in a local manifold structure formed by DPC.

The pseudo-code of the proposed DPCStr is described in Algorithm 3 which returns $next$. Note that $next$ is the set indicating space structure revealed by DPC. Specifically, the local density and the offset distance are calculated for each sample in X_{SSL} at Lines 1-5. Then, DPCStr makes each sample point to its nearest neighbor with a higher local density at Lines 6-9. Please note for Algorithm 3, several points need to be explained:

(a) At Line 2 and Line 5 of Algorithm 5, the proposed DPCstr helps STDPboost find high-confidence unlabeled samples. Compared to strategies for finding high-confidence unlabeled samples in existing self-training methods, DPCSt is parameter-free.

Algorithm 3 *DPCStr*

```
Input: $L$ (The set of $n_L$ labeled samples), $U$ (The set of $n_U$ unlabeled samples)      Time complexity
Output: $next$ (The set indicating space structure revealed by DPC)
1:  $X_{SSL} = L \cup U$;                                                                      $O(1)$
2:  for $\forall x_i \in X_{SSL}$                                                              $O(n)$
3:      Calculating local density $p(x_i)$ by formulas (6)-(8);                                $O(n^2)$
4:      Calculating the offset distance $\delta(x_i)$ by formula (9);                          $O(n^2)$
5:  end for                                                                                    —
6:  for $\forall x_i \in X_{SSL}$                                                              $O(n)$
7:      Finding the nearest neighbor $x_j$ of $x_i$ and $x_j$ is with a higher local density than $x_i$;   $O(nlogn)$
8:      $next(x_i) = x_j$;                                                                     $O(n)$
9:  end for                                                                                    —
10: return $next$;                                                                             $O(1)$
```

Algorithm 4 *Adaboost*$_{SEMI}$

```
Input: $L$ (The set of $n_L$ labeled samples), $M$ (The number of weak classifiers), $N$ (The number
of synthetic samples for each base sample in SMOTE), $k$ (The parameter of searching for $k$      Time complexity
nearest neighbors in SMOTE)
Output: $F$ (The trained ensemble classifier)
1:  $\forall x_i \in L, w_i = 1/n_L$;                                                          $O(n_L)$
2:  for $m = 1$ to $M$                                                                         $O(M)$
3:      Using $W$ to resample $L$, thus forming labeled sample set $L'$;                       $O(M \times n_L)$
4:      $X_{syn} = SMOTE(L', N, k)$; % using SMOTE to generate synthetic samples
        based on $L'$;                                                                         $O(M \times N \times n_L \times log(n_L))$
5:      $X_{training} = L' \cup X_{syn}$;                                                      $O(M)$
6:      Using $X_{training}$ to train a weak classifier $f_m$;                                 $O(M \times f)$
7:      Calculating the error rate $\varepsilon$ of $f_m$ by using formula (5);                $O(M \times f)$
8:      if $\varepsilon > 0.5$
9:          $a_m = 0$;                                                                         $O(M)$
10:         Reset $W$: $\forall\, x_i \in L, w_i = 1/n_L$;                                     $O(M \times n_L)$
11:     else if $\varepsilon == 0$
12:         $a_m = 10$;                                                                        $O(M)$
13:         Reset $W$: $\forall\, x_i \in L, w_i = 1/n_L$;                                     $O(M \times n_L)$
14:     else
15:         $a_m = (1/2) \times ln((1-\varepsilon)/\varepsilon)$;
16:         for each $x_i \in L$                                                               $O(M \times n_L)$
17:             if $f_m(x_i) = y_i$                                                            $O(M \times n_L)$
18:                 $w_i = w_i/(2 \times (1-\varepsilon))$;                                    $O(M \times n_L)$
19:             else
20:                 $w_i = w_i/2 \times \varepsilon$;                                          $O(M \times n_L)$
21:             end if
22:         end for
23:     Normalizing $W$;                                                                       $O(M \times n_L)$
24: end for
25: return $F$ by using formula (2);                                                           $O(M)$
```

(b) As analyzed in the column labeled "Time complexity", the time complexity of DPCStr is $O(n^2)$.

B. ADABOOST$_{SEMI}$

After finding high-confidence unlabeled samples by the proposed DPCStr, an improved Adaboost is proposed to construct an ensemble classifier used to predict found high-confidence unlabeled samples in STDPboost. The improved Adaboost is named Adaboost$_{\text{SEMI}}$ and is inspired by SMOTE [40] and Adaboost [31].

As described in Section III-C, Adaboost can construct an ensemble classifier by using an iterative process and combining multiple weak classifiers linearly. However, the effectiveness of the ensemble classifier constructed by Adaboost heavily depends on the diversity and accuracy of multiple weak classifiers. Due to the limited number and

Algorithm 5 *STDPboost*

```
Input: $L$ (The set of labeled data), $U$ (The set of unlabeled data), $N$ (The
number of synthetic samples for each labeled sample), $M$ (The number
of ensemble classifiers), $k$ (The neighbor parameter in Adaboost$_{\text{SEMI}}$)      Time complexity
Output: $C$ (The trained classifier)
 1:  $X_{SSL} = L \cup U$;                                                    $O(1)$
 2:  $next = DPCStr(L, U)$;                                                   $O(n^2)$
 3:  while $|U| \neq 0$                                                       $O(T)$
 4:      $F = Adaboost_{SEMI}(L, M, N, k)$;            $O(T) \times (O(M \times N \times n_L \times log(n_L)) + O(M \times f))$
 5:      Finding $X_{HCS}$ by Definition 1 and $next$;                        $O(T)$
 6:      Using the trained ensemble classifier $F$ to predict $X_{HCS}$,
         thus forming the set of newly predicted sample $L_{new}$;            $O(T) \times O(F)$
 7:      $L = L \cup L_{new}$;                                                $O(T)$
 8:      $U = U - X_{HCS}$;                                                   $O(T)$
 9:  end while                                                                —
10:  return a given classifier $C$ trained on the improved $L$;               $O(1)$
```

unrepresented distribution of the initial labeled data in SSC, it is hard to train multiple weak classifiers with sufficient diversity and high accuracy in Adaboost. Hence, Adaboost$_{\text{SEMI}}$ improves Adaboost by using SMOTE to create synthetic labeled samples in each iteration of Adaboost. In Adaboost$_{\text{SEMI}}$, synthetic labeled samples generated by SMOTE can improve the number and distribution of the initial labeled data, which enhances the diversity and accuracy of multiple weak classifiers trained on the initial labeled data.

The pseudo-code of the proposed Adaboost$_{\text{SEMI}}$ is described in Algorithm 4 which returns a trained ensemble classier F. Their main difference between Adaboost$_{\text{SEMI}}$ and Adaboost is that Adaboost$_{\text{SEMI}}$ uses SMOTE to create synthetic labeled samples at Line 4. Then, Adaboost$_{\text{SEMI}}$ uses synthetic labeled samples to improve the training set of each weak classifier in each iteration at Line 5. Next, Adaboost$_{\text{SEMI}}$ employs the idea of Adaboost to calculate the error rate ε, the weight a_m of each weak classifier f_m and the sample distribution W at Lines 6-24. Finally, an effective ensemble classifier can be obtained F at Line 25. Please note for Algorithm 4, several points need to be explained:

(a) SMOTE is originally used to generate synthetic minority class samples, where the set of minority class samples is used as input. In Adaboost$_{\text{SEMI}}$, because the labeled set L' formed by the sample distribution W is used as an input of SMOTE at Line 4, SMOTE generates synthetic labeled samples for each class, which improves the sample number and distribution of each class.

(b) As analyzed in the column labeled "Time complexity", the time complexity of Adaboost$_{\text{SEMI}}$ is $O(M \times N \times n_L \times log(n_L)) + O(M \times f)$, where M is the number of weak classifiers, N is the parameter in SMOTE, $O(f)$ is the time complexity of the adopted weak classifier and n_L is the sample number in L.

(c) The adopted weak classifier f_m is the same as the final trained classifier C in STDPboost. More specifically, if STDPboost is used to train a k nearest neighbor (KNN) [41] classifier, the KNN classifier is used as the weak classifier in Adaboost$_{\text{SEMI}}$.

(d) SMOTEBoost [33] also use SMOTE to improve Adaboost. Yet, SMOTEBoost is mainly used for imbalanced classification and aims to generate synthetic minority class samples. Compared to SMOTEBoost, the proposed Adaboost$_{\text{SEMI}}$ is mainly used for SSC. Adaboost$_{\text{SEMI}}$ can generate synthetic labeled samples to improve the insufficient labeled samples for each class, which make itself more suitable for SSC.

C. PSEUDO-CODE OF STDPboost

The pseudo-code of STDPboost is described in Algorithm 5. Line 2 shows the idea of the proposed DPCStr. Lines 3-9 show the iterative self-taught process of STDPboost: a) First, an ensemble classifier F is constructed by Adaboost$_{\text{SEMI}}$ on L at Line 4; b) Second, the proposed DPCStr is used to find the set of high-confidence unlabeled samples X_{HCS} Line 5; c) Third, the ensemble classifier F is used to predict the set of high-confidence unlabeled samples X_{HCS}, thus forming a newly predicted sample set L_{new} at Line 6; d) Four, L_{new} is added to L at Line 7 and the set of unlabeled data U is updated at Line 8. The above self-taught iteration stops until all unlabeled samples are predicted at Line 3. Finally, a given classifier C is trained on improved L and obtained at Line 10. Please note for Algorithm 5, several points need to be highlighted:

(a) Some existing self-training methods (e.g. STDP [23] and STDPCEWS [24]) also use DPC to find high-confidence unlabeled samples. Compared to them, DPCStr in STDPboost is parameter-free.

(b) Compared to existing self-training methods with ensemble classifiers [27], [28], [29], [30], [31], the ensemble classifier F constructed by the proposed Adaboost$_{\text{SEMI}}$ is more suitable for SSC since Adaboost$_{\text{SEMI}}$ can improve the sample number and distribution of the labeled set L by generating synthetic samples for each class.

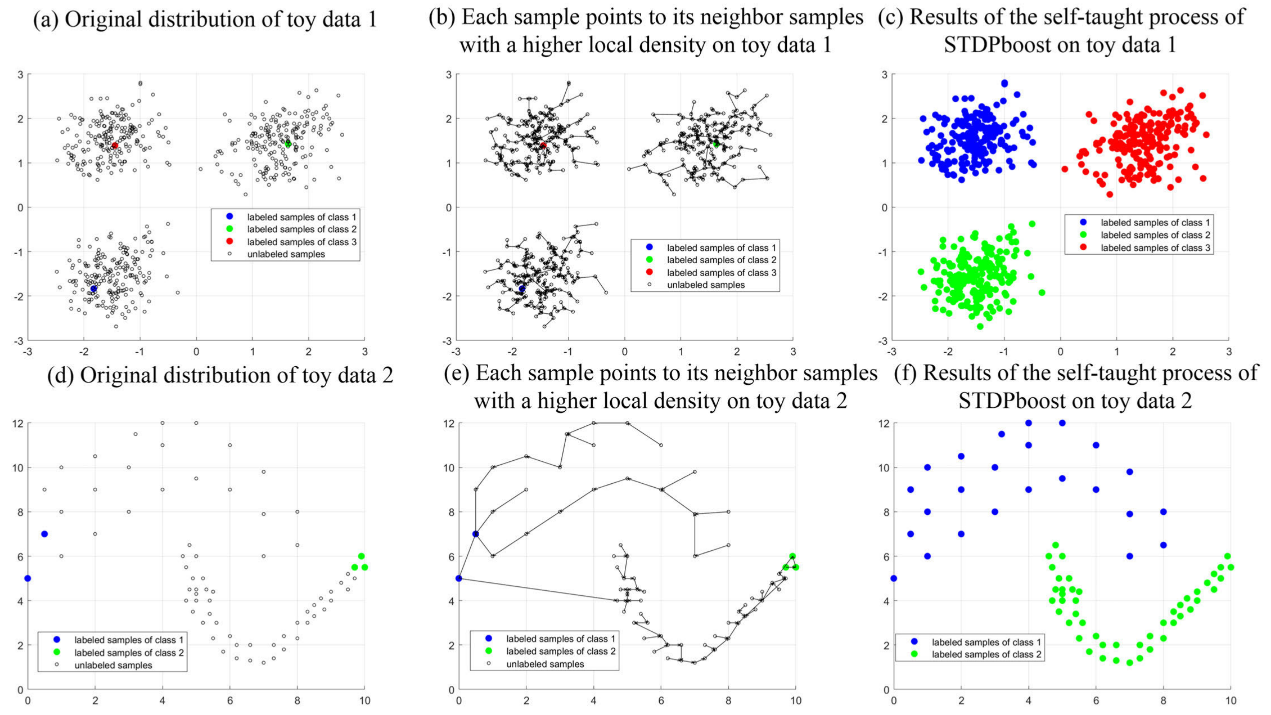

FIGURE 4. Results of the self-taught process of STDPboost on two toy data.

(c) As analyzed in the column labeled "Time complexity", the time complexity of STDPboost is $O(n^2)+O(T)\times(O(M\times N\times n_L\times log(n_L))+O(M\times f))+O(T)\times O(F)$, where M is the number of weak classifiers, N is the parameter in SMOTE, $O(f)$ is the time complexity of the adopted weak classifier, n_L is the sample number in L, $O(F)$ is the time complexity of the ensemble classifier F and T is the maximum number of iterations in STDPboost. The computational efficiency of STDPboost will be validated in Section V-D.

(d) Fig. 4 visualizes the results of the self-taught process of STDPboost in two toy data, where the CART (Classification And Regression Tree) classifier [42] is adopted in Adaboost$_{\text{SEMI}}$. It can be proved from Fig. 4 that STDPboost can correctly predict unlabeled samples on spherical or non-spherical data with a limited number and unrepresented distribution of the initial labeled data, which indirectly indicates the effectiveness of STDPboost.

V. EXPERIMENTS

Experiments are performed on a server that has a CPU of the Inter Core i7 with 3.10 GHz, a memory with 32GB, and a 64-bit Windows 10 operating system.

A. EXPERIMENTAL SETTINGS

Experimental benchmark data sets are selected from the machine learning open databases, such as UCI (https://www.uci.edu/) and Kaggle (https://www.uci.edu/). These benchmark data sets from UCI or Kaggle are often used to prove the effectiveness of machine-learning algorithms [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31].

To provide a clearer introduction to the experimental benchmark data sets, Table 1 describes experimental benchmark data sets in terms of data set name, sample number (#Examples), attribute number (#Attributes), class number (#Classes), and application areas. Observed from Table 1, the sample number of the experimental benchmark data sets ranges from 351 to 5000, the class number of the experimental benchmark data sets ranges from 2 to 7, and the attribute number of the experimental benchmark data sets ranges from 4 to 57. It can be seen from the column labeled "Application Areas" of Table 1 that experimental benchmark data sets come from 7 fields, such as medicine, business, computer security, physical, life, game, and bioinformatics. Generally, various benchmark data sets with different distributions are used to validate the effectiveness, robustness and practicality of the proposed STDPboost. Especially, challenging benchmark data sets (e. g. Cervical Cancer, Biodegradation, Wine Quality White, Ionosphere, and Wisconsin Diagnostic Breast Cancer) with relatively high dimensions, multiple classes, or/and relatively large sample sizes in frequently applied fields (e.g. medicine, business, and bioinformatics) are adopted.

As with the existing work [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], the inductive semi-supervised classification is focused on. The 10-fold cross-validation is used to divide each real data set into 10 parts, where the training set contains 9 parts and the test set contains 1 part. To simulate the semi-supervised learning environment, 10% of samples are labeled and 90% of samples are unlabeled in the training set of each data set. All

TABLE 1. Experimental real data sets.

ID	Data set name	#Examples	#Attributes	#Classes	Application Areas
1	Mammographic Masses	961	5	2	Medicine
2	Wine Quality Red	1599	11	6	Business
3	Phishing Websites	1353	9	3	Computer Security
4	Indian Liver Patient	583	10	2	Medicine
5	Cervical Cancer	858	35	2	Medicine
6	Australian Credit Approval	690	14	2	Physical
7	Vehicle	846	18	4	Life
8	Contraceptive Method Choice	1473	8	3	Medicine
9	Blood Transfusion Service Center	748	4	2	Medicine
10	Tic Tac Toe Endgame	958	9	2	Game
11	Wine Quality White	4898	11	7	Business
12	Biodegradation	1055	41	2	Medicine
13	Cardiotocography	2126	22	3	Life
14	Ionosphere	351	34	2	Physical
15	Wisconsin Diagnostic Breast Cancer	569	30	2	Medicine
16	Abalone	4177	8	3	Bioinformatics
17	Spambase	4501	57	2	Computer Security
18	WaveForm	5000	21	3	Physical

TABLE 2. The description of comparative self-labeled methods and base classifiers.

ID	Algorithms	Categories	Parameters
1	KNN	Trained base classifier	*k*=3
2	CART	Trained base classifier	The default parameters (i.e., *MinLeafSize*=1, *MinParentSize*=10 and *MaxNumSplits*=*n*-1) of Matlab2022
3	STSFCM	The self-training method based on the strategy for finding high-confidence unlabeled samples	Threshold ε =1/(the number of classes)
4	STDP	The self-training method based on the strategy for finding high-confidence unlabeled samples	P_a=2
5	STDPCEWS	The self-training method based on the strategy for finding high-confidence unlabeled samples	P_a=2 and $\alpha = 0.05$
6	NaNG-ST	The self-training method based on the strategy for finding high-confidence unlabeled samples	
7	Co-Adaboost	The self-training method based on ensemble classifiers	*Committee_members*=10, *Pool_U*=50 and *MAX_ITER*=100
8	Tri-training	The self-training method based on ensemble classifiers	
9	BoostSTIG	The self-training method based on ensemble classifiers	*f*=0.25, *M*=10 and P_a=2

experiments were repeated 10 times with the 10-fold cross-validation and the above strategy for partitioning labeled and unlabeled samples.

The average inductive classification accuracy (ACA) and the average running time of 10 executions are adopted as evaluation metrics. The ACA is formulated in formulas (11).

$$ACA = \frac{\sum_{i=1}^{10} \frac{|CorX_{test}^i|}{|X_{test}^i|}}{10} \tag{11}$$

In formula (11), X_{test}^i is the test set of *i*th experiment. $CorX_{test}^i$ is the set of samples correctly predicted by the trained classifier C in X_{test}^i. The related metrics of imbalanced classification (e.g. F-measure, G-mean) are not considered since our work mainly focuses on balanced classification.

To validate the effectiveness, popular base classifiers (i.e., trained classifiers) and state-of-the-art-self-labeled methods are adopted for comparison and are described in Table 2. Due to their popularity, stability and classicality, the KNN classifier [45] and the CART classifier [46] have been favored by scholars in image recognition, natural language processing, speech analysis, etc. Hence, KNN and CART are adopted as trained classifiers.

STSFCM [22], STDP [23], STDPCEWS [24], NaNG-ST [26], Co-Adaboost [28], Tri-training [29] and Boost-STIG [31] are adopted as comparative self-training methods because they have the same objective (i.e., overcoming mislabeling in the self-training method) as the proposed STDPboost. STSFCM, STDP, STDPCEWS and NaNG-ST

TABLE 3. ACA of comparative self-training methods in training KNN (%).

Data sets	STSFCM	STDP	STDPCEWS	NaNG-ST	Co-Adaboost	Tri-training	BoostSTIG	STDPboost
Mammographic Masses	78.35	79.34	80.19	79.38	78.42	80.45	80.11	**81.44**
Wine Quality Red	48.13	50.63	47.50	**52.50**	52.14	48.13	51.88	51.13
Phishing Websites	85.19	83.41	84.48	85.13	83.23	84.44	84.52	**85.48**
Indian Liver Patient	77.59	75.86	72.41	75.86	75.86	77.59	73.45	**78.62**
Cervical Cancer	88.37	87.22	87.67	88.41	88.53	88.53	86.05	**89.05**
Australian Credit Approval	67.67	69.57	69.57	68.87	68.87	65.22	**71.01**	69.57
Vehicle	62.41	64.88	64.35	64.32	**65.19**	62.82	63.82	63.65
Contraceptive Method Choice	48.92	**52.08**	51.35	48.77	47.41	49.32	48.65	49.32
Blood Transfusion Service Center	73.14	73.73	72.74	74.15	73.33	74.41	74.67	**75.15**
Tic Tac Toe Endgame	61.04	63.54	62.42	62.75	63.75	61.13	64.17	**65.50**
Wine Quality White	39.43	39.76	40.61	41.67	40.67	41.22	41.22	**42.44**
Biodegradation	71.92	72.81	72.87	72.32	71.32	71.58	71.53	**73.26**
Cardiotocography	83.02	82.55	81.13	79.25	79.25	**83.49**	81.64	82.42
Ionosphere	65.71	63.86	64.86	64.22	65.67	64.57	66.17	**66.77**
Wisconsin Diagnostic Breast Cancer	89.12	90.41	89.47	89.57	89.82	89.44	90.23	**90.96**
Abalone	44.63	45.22	44.11	43.30	43.30	45.34	46.04	**47.61**
Spambase	**69.57**	65.65	69.13	69.13	69.13	69.78	68.70	67.61
WaveForm	**82.11**	79.74	79.45	80.41	80.15	80.21	79.79	80.82
Average	68.68	68.90	68.57	68.89	68.67	68.76	69.09	**70.04**
Mean Rank	3.83	4.17	3.72	4.42	3.83	4.58	4.69	**6.75**
Nemenyi test	+	+	+	+	+	+	+	N/A

TABLE 4. ACA of comparative self-training methods in training cart (%).

Data sets	STSFCM	STDP	STDPCEWS	NaNG-ST	Co-Adaboost	Tri-training	BoostSTIG	STDPboost
Mammographic Masses	81.46	80.21	79.48	81.33	81.47	79.14	81.24	**82.13**
Wine Quality Red	43.13	43.13	42.50	42.50	42.50	46.88	46.25	**47.38**
Phishing Websites	83.09	81.62	82.35	82.35	82.35	76.47	84.56	**85.29**
Indian Liver Patient	**77.97**	76.27	77.69	77.82	76.79	76.58	74.27	75.44
Cervical Cancer	**93.81**	92.16	91.75	91.66	93.02	94.19	92.91	92.87
Australian Credit Approval	83.71	84.55	81.72	83.16	83.49	**84.07**	83.78	82.61
Vehicle	69.59	68.53	65.88	66.74	66.85	67.65	67.76	**69.88**
Contraceptive Method Choice	44.90	46.12	45.43	46.02	43.41	46.33	42.18	**46.94**
Blood Transfusion Service Center	87.84	90.54	90.19	89.47	89.55	88.63	**90.31**	90.23
Tic Tac Toe Endgame	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**	**66.67**
Wine Quality White	47.44	46.22	45.81	**46.81**	45.81	45.67	46.22	45.94
Biodegradation	75.24	**77.14**	75.24	75.24	75.24	76.86	76.10	75.38
Cardiotocography	90.85	90.39	91.08	91.08	91.08	91.79	**92.43**	92.31
Ionosphere	**77.14**	75.29	76.29	75.29	75.29	77.02	76.29	76.86
Wisconsin Diagnostic Breast Cancer	91.21	91.44	91.32	91.23	90.87	92.98	92.15	**93.74**
Abalone	47.72	48.84	47.08	47.82	47.19	48.81	**49.40**	47.88
Spambase	86.09	85.65	85.61	88.72	87.26	88.87	88.22	**89.78**
WaveForm	78.21	77.79	77.14	78.45	76.98	78.17	78.82	**79.11**
Average	73.67	73.47	72.96	73.46	73.10	73.71	73.86	**74.47**
Mean Rank	4.75	4.53	2.94	3.94	3.36	5.03	5.36	**6.08**
Nemenyi test	=	+	+	+	+	=	=	N/A

employ clustering-based or graph-based strategies for finding high-confidence unlabeled samples in each iteration to alleviate mislabeling. Co-Adaboost, Tri-training and BoostSTIG employ the ideas of ensemble classifiers to alleviate mislabeling. In BoostSTIG, Adaboost is adopted to predict unlabeled samples in each iteration of STDP. Parameters of the above comparative methods are set as their standard versions. The proposed STDPboost needs to set 3 parameters (i.e., M, N, and k). M is set to 10, N is set to 2 and k is set to 5 in experiments as some suggestions in the studies [31], [33].

B. COMPARING STDPboost WITH POPULAR SELF-TRAINING METHOD IN CLASSIFICATION ACCURACY

In order to prove the effectiveness of STDPboost, STDPboost is compared with 7 state-of-the-art self-training methods in training the KNN classifier and the CART classifier. The percentage of the initial labeled data is set to 10%, and others on the training set of each benchmark are unlabeled data. The empirical results of ACA for comparative methods in terms of the KNN classifier and the CART classifier are reported in Tables 3-4. The highest value of each row in Tables 3-4 is bold.

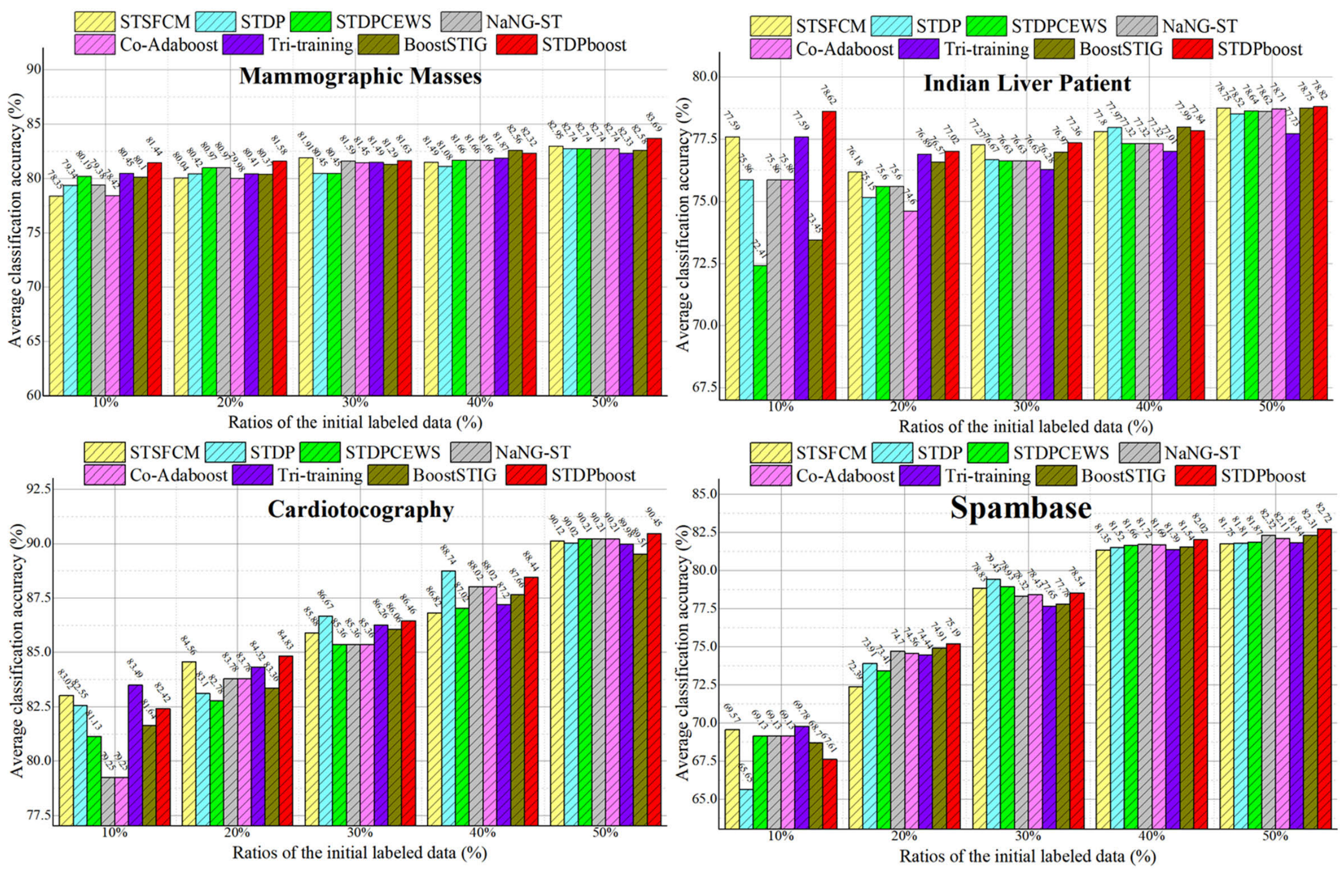

FIGURE 5. ACA of comparative self-training methods in training KNN classifier with different ratios of the initial labeled data.

STDPboost achieves the highest ACA for 11 of 18 data sets in Table 3, the highest average ACC for 9 of 18 data sets in Table 4. The row labeled "Average" indicates the average results of all data sets. Observing the row labeled "Average" of Tables 3-4, STDPboost also achieves the highest average results of all data sets in Tables 3-4. The above results prove that STDPboost outperforms comparative self-training methods in training the KNN classifier and the CART classifier on most data sets. Additionally, the ACA of STDPboost may be lower than that of comparative self-training methods on a few data sets. For example, STDP outperforms STDPboost on 3 data sets (Cardiotocography, Spambase and WaveForm) in Table 3. The reason may be that none of the self-training methods can be applied to all data sets due to the complex data distribution for various data sets. Despite all this, STDPboost can apply to and can be robust to more data sets than other comparative self-training methods because it can achieve the highest ACA on most data sets.

The results of Tables 3-4 may be incommensurable by averaging the ACA of all data sets [43]. Hence, the results of Tables 3-4 also are analyzed by the mean ranks of the Friedman test [43] in the row labeled "Mean Rank". The mean rank first sorts the results of each row (For instance, the sorting values of comparative self-training methods on the data set WaveForm in Table 3 are 8, 2, 1, 6, 4, 5, 3 and 7). Then, the "Mean Rank" averages the sorting values of all rows. A better algorithm can be with a larger mean rank. The row labeled "Mean Ranks" proves that STDPboost with the largest mean ranks is better than others.

The post-hoc Nemenyi test [43] with a significance level of 0.05 is also used to analyze the results in Tables 3-4. The row labeled "Nemenyi test" indicates the post-hoc Nemenyi test. In the row labeled "Nemenyi test". The symbol "+" indicates that STDPboost is significantly better than the comparative self-training method in the given column. The symbol "=" indicates that there is no significant difference between STDPboost and the comparative self-training method in the given column. The row labeled "Nemenyi test" have proven that STDPboost is statistically significantly superior to comparative self-training methods (e.g. STSFCM, STDP, STDPCEWS, NaNG-ST, Co-Adaboost, Tri-training and BoostSTIG) in training the KNN classifier and STDPboost is statistically significantly superior to most comparative self-training methods (e.g. STDP, STDPCEWS, NaNG-ST and Co-Adaboost) in training the CART classifier.

Additionally, the column labeled "Average" of Tables 3-4 has shown that STDPboost with the CART classifier can achieve a higher average classification than STDPboost with

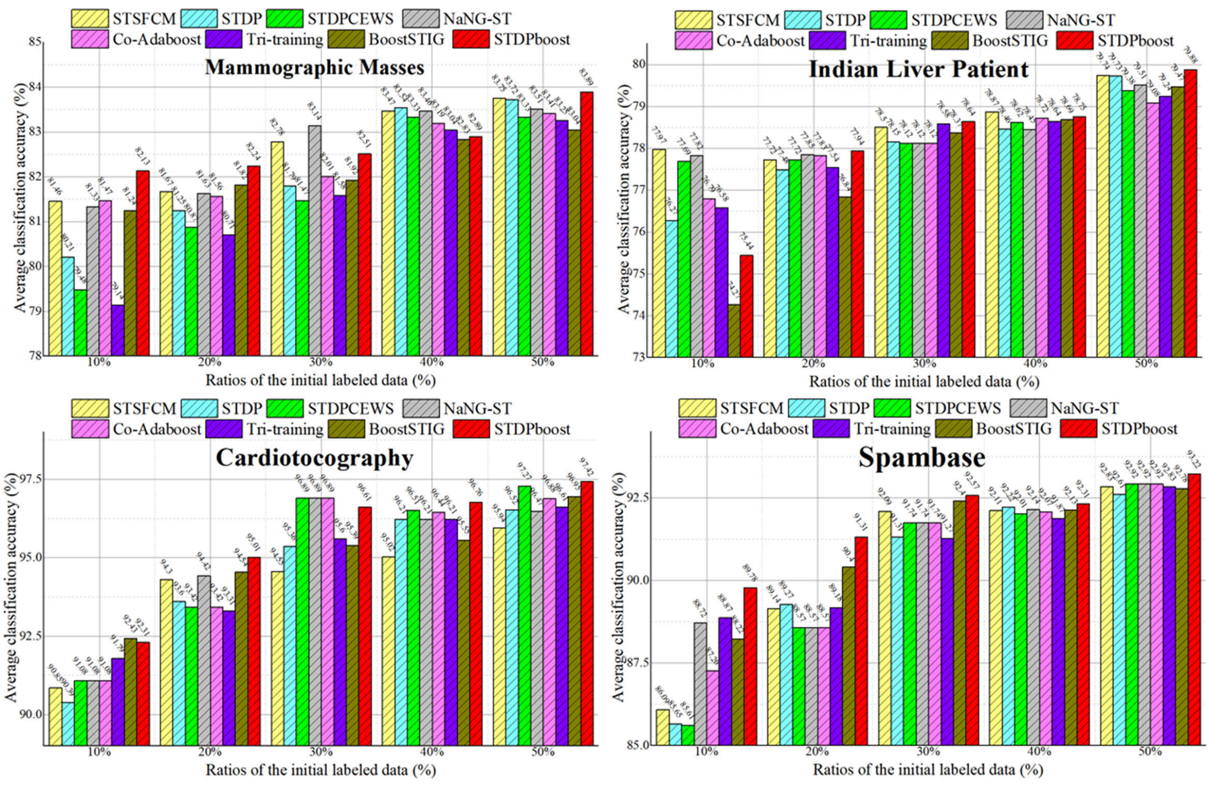

FIGURE 6. ACA of comparative self-training methods in training CART classifier with different ratios of the initial labeled data.

the KNN classifier. Hence, STDPboost is recommended to combine the CART classifier to complete SSC tasks.

Generally, Tables 3-4 can prove that STDPboost outperforms STSFCM, STDP, STDPCEWS, NaNG-ST, Co-Adaboost, Tri-training and BoostSTIG in training the KNN classifier and the CART classifier on most benchmark data sets, possibly due to the superiority of the proposed DPC-based strategy for finding high-confidence unlabeled samples and the proposed Adaboost$_{\text{SEMI}}$ in STDPboost.

C. EXPERIMENTS IN DISCUSSING THE INFLUENCE OF DIFFERENT RATIOS OF THE INITIAL LABELED DATA

To further illustrate the effectiveness of the proposed STDPboost, STDPboost is compared with 7 comparative self-training methods with different percentages of the initial labeled data. The results of ACA for comparison methods with different percentages of the initial labeled data are reported in Figs. 5-6, in which the percentage of the initial labeled data is increased from 10% to 50%. Due to time and resource constraints, representative 4 real benchmarks (Mammographic Masses, Indian Liver Patient, Cardiotocography and Spambase) are used in Figs. 5-6.

It is not difficult to find from Fig. 5 that a) STDPboost achieves the best ACA on the data set Mammographic Masses in training the KNN classifier when the percentages of labeled samples are 10%, 20% and 50%; b) STDPboost achieves the best ACA on the data set Indian Liver Patient in training the KNN classifier when the percentages of labeled samples are 10%, 20%, 30% and 50%; c) STDPboost achieve the best ACA on the data set Cardiotocography in training the KNN classifier when the percentages of labeled samples are 20% and 50%; d) STDPboost achieve the best ACA on the data set Spambase in training the KNN classifier when the percentages of labeled samples are 20%, 40% and 50%;

Additionally, it is not difficult to find from Fig. 6 that a) STDPboost achieves the best ACA on the data set Mammographic Masses in training the CART classifier when the percentages of labeled samples are 10%, 20%and 50%; b) STDPboost achieves the best ACA on the data set Indian Liver Patient in training the CART classifier when the percentages of labeled samples are 20%, 30% and 50%; c) STDPboost achieve the best ACA on the data set Cardiotocography in training the CART classifier when the percentages of labeled samples are 20%, 40% and 50%; d) STDPboost achieve the best ACA on the data set Spambase in training the CART classifier when the percentages of labeled samples are 10%-50%;

With the increase in the proportion of the initial labeled data, the initial labeled data can better approach the real distribution of the original data. If so, comparative self-training methods may more accurately estimate the confidence of unlabeled samples and make a more accurate prediction for unlabeled samples. Hence, as the percentage of labeled data increases from 10% to 50%, the overall performance of ACA for all comparative self-training methods will be better in Figs. 5-6.

TABLE 5. **Average running time of comparative self-training methods (seconds).**

Data sets	STSFCM	STDP	STDPCEWS	NaNG-ST	Co-Adaboost	Tri-training	BoostSTIG	STDPboost
Mammographic Masses	0.43	0.50	1.97	0.21	0.25	1.49	2.49	1.86
Wine Quality Red	0.57	0.77	3.47	0.42	0.48	2.30	4.06	2.78
Phishing Websites	0.43	0.48	2.16	0.16	0.45	1.91	2.85	1.82
Indian Liver Patient	0.39	0.52	2.85	0.25	0.41	1.24	3.03	2.30
Cervical Cancer	0.40	0.63	5.37	0.37	0.44	2.01	5.30	4.16
Australian Credit Approval	0.41	0.62	2.26	0.27	0.47	1.34	1.92	1.61
Vehicle	0.45	0.58	4.27	0.28	0.48	1.39	4.04	3.83
Contraceptive Method Choice	0.41	0.57	5.31	0.36	0.45	2.07	5.14	4.93
Blood Transfusion Service Center	0.44	0.51	2.35	0.18	0.51	1.31	2.33	2.27
Tic Tac Toe Endgame	0.50	0.44	2.49	0.10	0.62	1.53	2.53	2.45
Wine Quality White	0.94	1.94	11.58	1.60	1.56	28.84	15.94	10.65
Biodegradation	0.39	0.59	5.29	0.25	0.49	1.66	5.69	4.56
Cardiotocography	0.45	0.91	9.63	0.60	0.65	3.73	9.78	7.49
Ionosphere	0.39	0.45	5.75	0.15	0.56	2.13	7.43	5.28
Wisconsin Diagnostic Breast Cancer	0.41	0.55	2.26	0.26	0.52	1.27	4.04	2.75
Abalone	0.45	2.30	17.44	1.56	2.56	25.31	18.30	16.34
Spambase	0.81	2.67	12.01	1.96	2.08	27.61	11.58	11.90
WaveForm	0.69	1.57	10.26	1.21	1.22	26.55	14.97	8.08

Generally, Figs. 5-6 can prove that STDPboost outperforms 7 comparative self-training methods with various percentages (from 10% to 50%) of the initial labeled data.

D. EXPERIMENTS IN VALIDATING THE COMPUTATIONAL EFFICIENCY

STDPboost is compared with 7 state-of-the-art self-training methods in the average running time of 10 executions. The CART is used as the trained classifier in STDPboost and comparative self-training methods. Table 5 reports the average running time of STDPboost and 7 comparative self-training methods. Observing Table 5, although STDPboost has no absolute advantage in average running time, STDPboost is still faster than STDPCEWS and BoostSTIG on most data sets.

VI. CONCLUSIONS AND PLANS

Mislabeling is one of the main challenges for self-training methods. Existing self-training variations overcoming mislabeling from one of two main aspects: a) using heuristic rules to find high-confidence unlabeled samples; b) enhancing prediction performance by employing ensemble classifiers. Yet, they still have the following shortcomings: a) strategies for finding high-confidence unlabeled samples heavily rely on parameters; b) employed ensemble classifiers originally designed for supervised classifiers and may not be suitable for semi-supervised classification due to the limited number and unrepresented distribution of the initial labeled data; c) few can overcome mislabeling from the above two aspects at the same time. To overcome the above issues, a novel self-training method (named STDPboost) based on density peaks clustering and improved Adaboost is proposed.

The main ideas of STDPboost include the following iterative self-taught process: a) An ensemble classifier is trained on the set of labeled data by using the proposed $\text{Adaboost}_{\text{SEMI}}$; b) the proposed DPCStr is used to find high-confidence unlabeled samples from the set of unlabeled data; c) the trained ensemble classifier is used to predict high-confidence unlabeled samples; d) predicted high-confidence samples with pseudo labels are added to the set of labeled data. The above iterative self-taught process repeats until all unlabeled samples are predicted. After that, STDPboost outputs a given classifier trained on the improved set of labeled data.

The main advantages of the proposed STDPboost are concluded as follows: a) it is parameter-free; b) it can overcome mislabeling by a new parameter-free strategy (DPCStr) for finding high-confidence unlabeled samples and a new ensemble classifier ($\text{Adaboost}_{\text{SEMI}}$) more suitable for SSC.

The main contributions are highlighted as follows:

(a) A new self-training method named STDPboost is proposed.

(b) A new parameter-free DPC-based strategy (DPCStr) for finding high-confidence unlabeled samples in STDPboost is proposed.

(c) A new ensemble method named $\text{Adaboost}_{\text{SEMI}}$ for constructing an ensemble classifier in STDPboost is proposed.

(d) Intensive experimental results with extensive real benchmarks, 2 trained classifiers (KNN and CART), 7 state-of-the-art self-training methods (STSFCM, STDP, STDPCEWS, NaNG-ST, Co-Adaboost, Tri-training and BoostSTIG) are reported.

Empirical experiments with various benchmark data sets from UCI and Kaggle comprehensively conclude that a) STDPboost outperforms 7 state-of-the-art self-training methods on most benchmark data sets with various percentages of the initial labeled data; b) STDPboost is faster than STDPCEWS and BoostSTIG.

In plans, we are interested in overcoming the defect of the parameter selection (N, M and k) in the proposed STDPboost. We are interested improve STDPboost to make it suitable for class-imbalanced data sets and consider more evaluation metrics (e.g. F-measure and G-mean) for imbalanced classification. We are also interested to apply the proposed STDPboost to more non-real-time practical applications, such as intelligent medical detection, financial risk control, and intelligent management decision-making.

REFERENCES

[1] Y. Wang, Y. Y. Tang, L. Li, H. Chen, and J. Pan, "Atomic representation-based classification: Theory, algorithm, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 6–19, Jan. 2019.

[2] C. Ji, Y. Wang, Z. Gao, L. Li, J. Ni, and C. Zheng, "A semi-supervised learning method for MiRNA-disease association prediction based on variational autoencoder," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 19, no. 4, pp. 2049–2059, Jul. 2022.

[3] Y. Wang, W. Li, Z. Huang, R. Tao, and P. Ma, "Low-slow-small target tracking using relocalization module," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[4] S. Gupta and V. Kant, "Credibility score based multi-criteria recommender system," *Knowl.-Based Syst.*, vol. 196, May 2020, Art. no. 105756.

[5] J. Zhang, Y. Zhou, and C. Zong, "Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 10, pp. 1842–1853, Oct. 2016.

[6] Q. Liu, L. Yu, L. Luo, Q. Dou, and P. A. Heng, "Semi-supervised medical image classification with relation-driven self-ensembling model," *IEEE Trans. Med. Imag.*, vol. 39, no. 11, pp. 3429–3440, Nov. 2020.

[7] Y. Sun, S. Ding, L. Guo, and Z. Zhang, "Hypergraph regularized semi-supervised support vector machine," *Inf. Sci.*, vol. 591, pp. 400–421, Apr. 2022.

[8] W. P. Amorim, A. X. Falcão, and J. P. Papa, "Multi-label semi-supervised classification through optimum-path forest," *Inf. Sci.*, vol. 465, pp. 86–104, Oct. 2018.

[9] Q. Zhang, T. Chu, and C. Zhang, "Semi-supervised graph based embedding with non-convex sparse coding techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2193–2207, May 2021.

[10] J. Li, M. Zhou, Q. Zhu, and Q. Wu, "A framework based on local cores and synthetic examples generation for self-labeled semi-supervised classification," *Pattern Recognit.*, vol. 134, Feb. 2023, Art. no. 109060.

[11] S. Zheng and J. Zhao, "A self-adaptive temporal-spatial self-training algorithm for semisupervised fault diagnosis of industrial processes," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6700–6711, Oct. 2022.

[12] H. Cui, G. Wang, Y. Li, and R. E. Welsch, "Self-training method based on GCN for semi-supervised short text classification," *Inf. Sci.*, vol. 611, pp. 18–29, Sep. 2022.

[13] X. Wang, H. Chen, H. Xiang, H. Lin, X. Lin, and P.-A. Heng, "Deep virtual adversarial self-training with consistency regularization for semi-supervised medical image classification," *Med. Image Anal.*, vol. 70, May 2021, Art. no. 102010.

[14] R. Chen, Y. Ma, L. Liu, N. Chen, Z. Cui, G. Wei, and W. Wang, "Semi-supervised anatomical landmark detection via shape-regulated self-training," *Neurocomputing*, vol. 471, pp. 335–345, Jan. 2022.

[15] Y. Xia, D. Yang, Z. Yu, F. Liu, J. Cai, L. Yu, Z. Zhu, D. Xu, A. Yuille, and H. Roth, "Uncertainty-aware multi-view co-training for semi-supervised medical image segmentation and domain adaptation," *Med. Image Anal.*, vol. 65, Oct. 2020, Art. no. 101766.

[16] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. Kwoh, X. Li, and C. Guan, "ADAST: Attentive cross-domain EEG-based sleep staging framework with iterative self-training," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 1, pp. 210–221, Feb. 2023.

[17] J. Li, B. Sun, S. Li, and X. Kang, "Semisupervised semantic segmentation of remote sensing images with consistency self-training," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5615811.

[18] M. Li and Z.-H. Zhou, "SETRED: Self-training with editing," in *Proc. Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, Jan. 2005, pp. 611–621.

[19] Y. Wang, X. Xu, H. Zhao, and Z. Hua, "Semi-supervised learning based on nearest neighbor rule and cut edges," *Knowl.-Based Syst.*, vol. 23, no. 6, pp. 547–554, Aug. 2010.

[20] Z. Wei, H. Wang, and R. Zhao, "Semi-supervised multi-label image classification based on nearest neighbor editing," *Neurocomputing*, vol. 119, pp. 462–468, Nov. 2013.

[21] M. M. Adankon and M. Cheriet, "Help-training for semi-supervised support vector machines," *Pattern Recognit.*, vol. 44, no. 9, pp. 2220–2230, Sep. 2011.

[22] H. Gan, N. Sang, R. Huang, X. Tong, and Z. Dan, "Using clustering analysis to improve semi-supervised classification," *Neurocomputing*, vol. 101, pp. 290–298, Feb. 2013.

[23] D. Wu, M. Shang, X. Luo, J. Xu, H. Yan, W. Deng, and G. Wang, "Self-training semi-supervised classification based on density peaks of data," *Neurocomputing*, vol. 275, pp. 180–191, Jan. 2018.

[24] D. Wei, Y. Yang, and H. Qiu, "Improving self-training with density peaks of data and cut edge weight statistic," *Soft Comput.*, vol. 24, no. 20, pp. 15595–15610, Oct. 2020.

[25] J. Li and Q. Zhu, "Semi-supervised self-training method based on an optimum-path forest," *IEEE Access*, vol. 7, pp. 36388–36399, 2019.

[26] J. Li, "NaNG-ST: A natural neighborhood graph-based self-training method for semi-supervised classification," *Neurocomputing*, vol. 514, pp. 268–284, Sep. 2022.

[27] I. Triguero, S. García, and F. Herrera, "SEG-SSC: A framework based on synthetic examples generation for self-labeled semi-supervised classification," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 622–634, Apr. 2015.

[28] J. Li, Q. Zhu, Q. Wu, and D. Cheng, "An effective framework based on local cores for self-labeled semi-supervised classification," *Knowl.-Based Syst.*, vol. 197, Jun. 2020, Art. no. 105804.

[29] S. Wang, Y. Guo, W. Hua, X. Liu, G. Song, B. Hou, and L. Jiao, "Semi-supervised PolSAR image classification based on improved tri-training with a minimum spanning tree," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8583–8597, Dec. 2020.

[30] S. Gu and Y. Jin, "Multi-train: A semi-supervised heterogeneous ensemble classifier," *Neurocomputing*, vol. 249, pp. 202–211, Aug. 2017.

[31] J. Li and Q. Zhu, "A boosting self-training framework based on instance generation with natural neighbors for k nearest neighbor," *Appl. Intell.*, vol. 50, no. 11, pp. 3535–3553, Nov. 2020.

[32] J. Chen and P. S. Yu, "A domain adaptive density clustering algorithm for data with varying density distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2310–2321, Jun. 2021.

[33] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003*. Berlin, Germany: Springer, Sep. 2003, pp. 107–119.

[34] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *Proc. 33rd Annu. Meeting Assoc. Comput. Linguistics*, 1995, pp. 189–199.

[35] Q. Xue, Y. Zhu, and J. Wang, "Joint distribution estimation and Naïve Bayes classification under local differential privacy," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 4, pp. 2053–2063, Oct. 2021.

[36] D. S. Mai, L. T. Ngo, L. H. Trinh, and H. Hagras, "A hybrid interval type-2 semi-supervised possibilistic fuzzy c-means clustering and particle swarm optimization for satellite image analysis," *Inf. Sci.*, vol. 548, pp. 398–422, Feb. 2021.

[37] Q. Zhu, J. Feng, and J. Huang, "Natural neighbor: A self-adaptive neighborhood method without parameter K," *Pattern Recognit. Lett.*, vol. 80, pp. 30–36, Sep. 2016.

[38] G. Ngo, R. Beard, and R. Chandra, "Evolutionary bagging for ensemble learning," *Neurocomputing*, vol. 510, pp. 1–14, Oct. 2022.

[39] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, and H. Yan, "A highly accurate framework for self-labeled semisupervised classification in industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 909–920, Mar. 2018.

[40] J. Li, Q. Zhu, Q. Wu, and Z. Fan, "A novel oversampling technique for class-imbalanced learning based on SMOTE and natural neighbors," *Inf. Sci.*, vol. 565, pp. 438–455, Jul. 2021.

[41] J. A. Romero-del-Castillo, M. Mendoza-Hurtado, D. Ortiz-Boyer, and N. García-Pedrajas, "Local-based k values for multi-label k-nearest neighbors rule," *Eng. Appl. Artif. Intell.*, vol. 116, Nov. 2022, Art. no. 105487.

[42] Y. Xu, X. Hu, J. Wei, H. Yang, and K. Li, "VF-CART: A communication-efficient vertical federated framework for the CART algorithm," *J. King Saud Univ., Comput. Inf. Sci.*, vol. 35, no. 1, pp. 237–249, Jan. 2023.

[43] P. Griffiths and J. Needleman, "Statistical significance testing and p-values: Defending the indefensible? A discussion paper and position statement," *Int. J. Nursing Stud.*, vol. 99, Nov. 2019, Art. no. 103384.